昭和54年5月2日第3種郵便物認可 ISSN 0387-9569
平成28年10月1日発行（毎月1回1日発行） 第42巻 第10号 通巻472号

コンピュータ・サイエンス&テクノロジ専門誌

Interface
インターフェース

データ解析時代の新定番
ライブラリ101付き Python



夏の2大定番入門講座

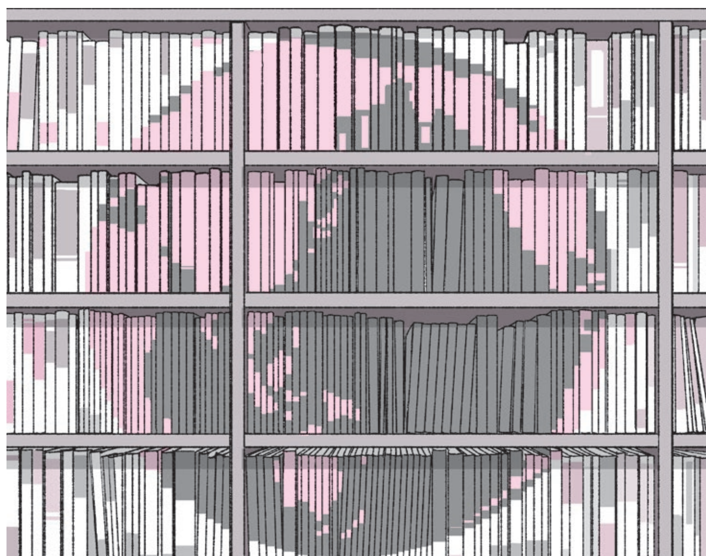
2016 10

全コンピュータ制覇！最近はWindowsにまでキテるらしい

超定番シェル×IoT

Interface 2016 10

Contents



DL 本誌のページ等でダウンロード・データを公開中です
http://interface.cqpub.co.jp/

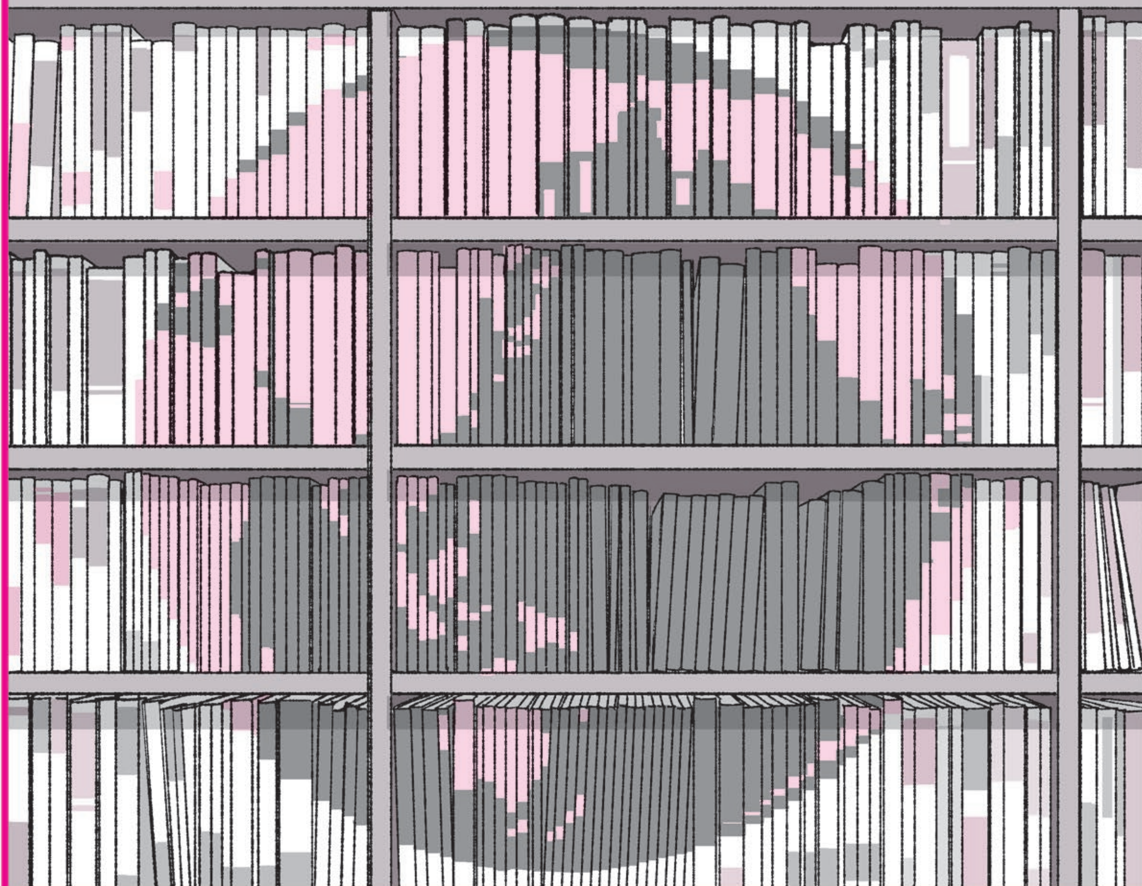
Interface 2016 10

本誌公式Twitter：@if_cq

Contents

お知らせ 連載「ラズパイ式走るリモート探査カメラ」「安全に使い回す！車載ソフトウェアの世界」はお休みします．連載「高速ワンチップ・マイコンではじめるソフトウェア無線」は終了します．第1特集 第5章が最終回（番外編）に相当します．

夏の2大定番入門講座



第1特集

データ解析時代の

新定番Python（パイソン）

オフ会 「Python＆シェル探検隊（仮）」

日時：2016年10月18日（火）19～21時
場所：東京巣鴨（CQ出版社）　会費：1,000円
詳細は本誌ウェブ・サイト（http://interface.cqpub.co.jp/）

第1章　初心者からプロまで使える科学技術計算の新定番

IoT時代にPythonをオススメする理由

佐藤 亮平

● その1：文法がシンプルで初心者にも優しい

Pythonはシンプルな文法の言語であり，プログラムを書くために必要な知識が少なくてすむことが特徴です．初めてプログラミング言語に触れる人にも分かりやすく，また，しばらくプログラミングを行っていない人が感覚を取り戻す際にも，さほど苦労はしないですみます．

Pythonは公式のドキュメントが簡潔で分かりやすく整備されています．日本語訳も同様です．Pythonを使っている人が周りにおらず，気軽に聞くことができない場合も，チュートリアル(**図1**)をはじめとしたドキュメントを読みながら学習すれば，だいぶ理解も早まるのではと思います．

```
http://docs.python.jp/2.7/tutorial/
index.html
```

● その2：オブジェクト指向対応で中上級者になっても使い続けられる

Pythonという言語の大きな利点の一つとして，初心者から熟練者まで，それぞれのやり方で書きやすいような文法ということがあります．つまり，プログラムを書くために最低限必要な知識はBASICのように少ない一方，Pythonはオブジェクト指向で開発を行うための言語機能がそろっていて，複雑で大規模なコードを書くことも困難ではありません．



図1　Pythonチュートリアル
Pythonの使い方に関するさまざまなドキュメントがここに掲載されている

さらに重要なこととして，文法がうまく練られているため，より良い書き方を覚えるために，一度に多くの時間を費やして学習する必要がありません．自分のペースで少しずつ覚えたことを使って，少しずつ自分のコードを良くできます．

ソフトウェア開発自体が仕事ではなく，あくまでやりたいことを実現する手段の一部としてプログラムを書いている人の多くは，学習の時間を集中的に取れないことも多いかと思います．Pythonのような言語は，そのような人にとってもお勧めできる言語です．

● その3：対話型環境が標準で用意されていて楽ちん

近年，言語の基本的な機能を簡単に試してみたい場合に，環境をインストールせずに気軽に行える仕組みが増えてきています．**図2**に示すようなIdeone (`https://ideone.com`)，repl.it (`https://repl.it`) などの優良なオンラインIDEサービスがあり，基本的な機能は無償・無登録で利用することができます．ウェブ・ブラウザを立ち上げるだけで，多くの言語のコードを書き，その場で動作を確かめることができます．Java，C/C++，Rubyなど，他のメジャーな言語の多くもそろっています．とはいえ，このようなサービスでは，ライブラリは標準的なものしか用意されていない場合が多く，また実行の本体はウェブ上にあるわけなので，ハードウェアに関係するような処理を実装する場合には，手元の環境でプログラムを開発する必要があるでしょう．

ところが，同様のことを自分のPC上で行おうとした場合，標準機能として対話環境が用意されていないことも多く，また実行前にコンパイルを実施する必要がある言語の多くは，コンパイル時間がネックとなって少々待たされることにもなりがちです．

Lisp系言語，Rubyなどとともに，Pythonには REPL (Read-Eval-Print-Loop) と呼ばれる高度な対話型環境が標準で用意されています．つまりプログラムをファイルに書き出さずとも，あたかも電卓を使って一つ一つの計算を確かめることができるように，1行

だけコードを書いて動作を確かめたり，関数を一つ書いて動作を確かめたりすることが標準の機能として可能です．

対話的に部品の動作を確かめながら，プログラムを積み上げていくことができるため，手間を増やさずに，きちんと動作するという安心感を持って開発ができます．

環境の立ち上げ方もシェルやコマンド・プロンプト上で，下記のように引き数なしでコマンドを呼び出すだけの簡単なものとなっています．

```
> Python
```

● その4：インデントが文法として決められていてコードが読みやすい

他の言語をすでに知っている人にとって，Pythonのコードを眺めたとき，見た目として一番目を引く特徴は，おそらく処理の範囲を示す中括弧{ }やbegin～endが存在しないことです．

Pythonは，処理のブロック（範囲）がインデント（字下げ）のみで表現されているため，そのような記述をする必要がなく，またそもそもそのように記述することができません．

Pythonに限らず多くの言語で，熟練者はコードにインデントを付けるようになります．インデントを付けると，処理のブロックがざっと眺めるだけで理解できて読みやすいからなのですが，逆に言えば，間違ったインデントがなされたコードを読む場合には，非常にイライラしますし，処理のブロックを勘違いしたため，間違った箇所にコードを追加して，バグを生み出す恐れもあります．

また，任意にインデントできる言語では，インデントの入れ方にしばしば複数の流儀があるため，別の流儀で書かれたコードを読む際には，その流儀に慣れるまでに少々時間がかかってしまいがちです．

Pythonでは，インデントの付け方が，マナーやローカル・ルールでなく文法として統一されているため，他人の書いたコードに違和感を感じることなく読むことができます．

なお，Python 環境に付属しているIDLEをはじめ，多くのテキスト・エディタや統合開発環境には，Pythonコードに対し，自動で適切な字下げを行う機能がすでに用意されています．スペース・キーやタブ・キーを連打する必要はありません．

● その5：科学技術計算が得意！ IoTに必要なライブラリが揃っている

Python，Ruby，Perlなどのスクリプト言語は，比較的データ加工に長けていることが特徴です．つまり，周辺のソフトウェア・ハードウェア環境からの情報を加工・処理した上で他の環境に出力するコードを書きやすいのです．例えば，IoT（Internet of Things）を試してみようと思えば，

- モノに付属のセンサからの情報を読み取る
- データを加工する
- 他のネット上のデバイスと自動で通信を行う

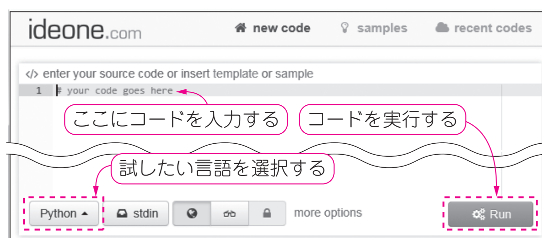
などが必要になります．

センサからの情報取得については，Pythonのみでは直接センサを制御するコードこそ書けないものの，C言語で書かれたプログラムとの連携によって実現できます．例えばArduinoを経由したシリアル通信を行うことにするなら，pySerialライブラリを用いて，さまざまなハードウェアにアクセスできます．あるいは，ラズベリー・パイでは，GPIOポートを経由してより直接的に制御するライブラリも用意されています．

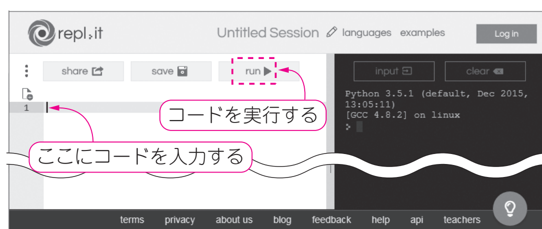
データ加工については，簡単な統計から機械学習まで，何百ものライブラリが用意されています．例えば，ディープ・ラーニングに限定しても，Caffe，Chainer，Lasagneなど，数多くの選択肢があります．画像加工についても，OpenCVをはじめとして，多くの画像処理ライブラリが用意されています．さらに，ただデータを加工するだけではなく，Matplotlib，ggplot，Bokehなどのライブラリを用いれば，それを分かりやすくビジュアル化させることもできます．

インターネット上の通信についても，標準のsocketライブラリで基本的な通信ができる上，本格的なウェブ・アプリにしたければ，Djangoなどのフレームワークを用いて開発することもできます．

以上で述べた多くのことは，PerlやRubyにもでき



（a）Ideone（https://ideone.com）



（b）repl.it（https://repl.it）

図2　オンラインIDEサービス
環境をインストールせずに言語の基本的な機能を試せる

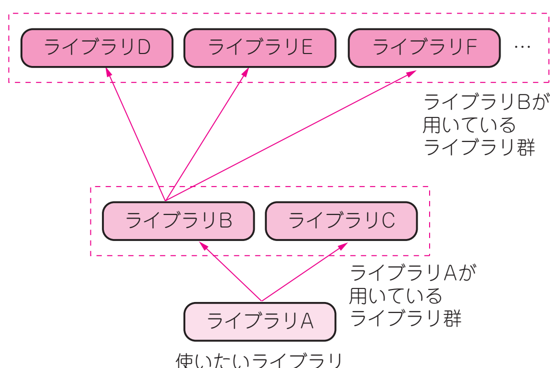


図3　新しいライブラリは既存ライブラリを使って簡単に作れる
依存関係を解決してインストールしてくれるパッケージ管理ソフトウェアが用意されている

るのですが，相対的に見て，その中でもPythonが比較的得意とするのは，数値計算，最適化，機械学習といった科学技術計算の分野です．

● その６：スクリプトなのに計算が高速

Pythonが科学技術計算を得意とする理由として，SciPyとNumPyという数値計算ライブラリの存在は外せません．他にも，表として表現できるデータに関してはPandasという重要なライブラリもあります．近年増えたライブラリの多くが，SciPy/NumPy上で計算を行っています．

スクリプト言語のデメリットとしてよく挙げられるのは，処理速度がCなどの言語に比べて遅いことです．Pythonの場合，一旦処理をSciPy/NumPyに渡してしまうと，機械語でコンパイルされたプログラムで計算がなされるため，Cなどで書かれたプログラムに対しても，見劣りすることのない速度で処理ができます．

さらに重要なこととして，科学技術計算のためのデータの受け渡しの形式として，SciPy，NumPy，Pandasなどのデータ型がデファクト・スタンダードの位置を占めていることがあります．つまり，これらライブラリで定義されているデータ型をいわば共通通貨として，さまざまなライブラリを連携させることができます．

例えばOpenCVをPython上で用いる場合には，OpenCVが内部処理としてはSciPy/NumPyの上に構築されているわけではないにもかかわらず，画像はNumPyのndarray型で受け渡すようになっています．画像をChainerに入力してディープ・ラーニングをさせるのもよいですし，他の処理もさせられます．

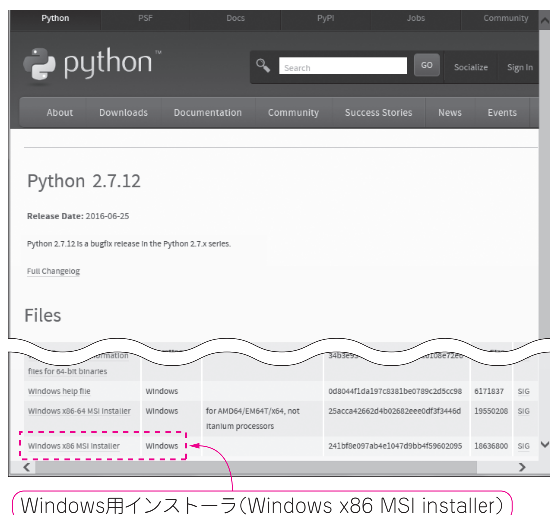


図4　Pythonの公式サイト
Windows用のPythonのインストーラが無償でダウンロードできる

● その７：ライブラリのインストールも簡単

RubyやPerlなどの言語には，それぞれの言語で書かれた膨大な数のライブラリをインストール，管理するためのパッケージ管理システム（Rubyにはgem，PerlにはCPAN）があります．

パッケージ管理システムはライブラリを導入するために必須の仕組みです．既存のライブラリの機能を活用して，新しい機能を付け加えるだけでライブラリを構築できるのは，ライブラリを作る側にとってはとても便利なことなのですが，逆にライブラリを使う側にとっては大変面倒なことであるともいえます．**図3**に示すように，あるライブラリAを使いたい場合，ライブラリAが用いているライブラリBとC，さらにライブラリBが用いているライブラリD，E，F…についてもインストールを行わなければばらないことを意味するからです．

インストールだけではなく，ただどれかのライブラリをアップデートする場合においても同様のことがいえるため，手作業でライブラリのインストールやアップデートを行おうとすると大変面倒な作業となります．パッケージ管理システムは，これら複雑なライブラリ間の依存関係を自動的に解決し，必要なライブラリをすべてインストールしてくれるので，ライブラリを導入する側では，使いたいライブラリが何に依存しているかを意識する必要がなくなるのです．

最新版のPythonには，標準でpipというバージョン管理ツールが付属しているため，このpipを用いて，前の項で挙げた多彩なライブラリ群を簡単にインストールできるのです．

また，何か新しいライブラリを作ったなら，このpipの仕組みに合わせ，適切なレポジトリ（PyPIなど）に公開することによって，全世界の人とその成果を共有することもできるのです．

● その8：無償である

Windowsなら，図4に示す公式サイトから，“Windows x86 MSI installer”をダウンロードし，インストールすれば使えるようになります．

https://www.python.org/downloads/release/python-2711/

お金を払ったり登録する必要はありません．多くのライブラリも同様に，無償，無登録で使うことができます．ただし，特に商用で使う場合には，言語，ライブラリそれぞれのライセンスを遵守していることのチェックが必要です．

また，Linuxを使っているなら，多くのディストリビューションにはPythonがすでに含まれていますから，シェル上で呼び出すだけで使えるようになっています．

注意すべき点は，Pythonには文法に互換性のない二つの系統，2.*x*系(2016年7月現在，バージョン2.7.12が最新)と，3.*x*系(2016年7月現在，バージョン3.5.2が最新)があることです．

より新しい仕様である3.*x*系の方が文法が洗練されたものとなっており，また，日本語を扱う際に重要なUnicode文字列の取り扱い方に改善が見られるため，より使いやすくなっています．

しかし，標準以外のライブラリの中には，いまだに3.*x*系に対応していないものもあります．逆に3.*x*系でしか動かないライブラリは少ないことから，どちらのバージョンを使うか悩んだ際は，バージョン2.*x*系を使うことをお勧めします．互換性がないとはいえ，別言語と感じるほどの違いではありませんから，片方の系統で覚えた知識はもう一方でも使えることでしょう．また同じマシンに2.*x*系と3.*x*系を共存させることもできます．

どちらの系統を使うか決めた場合，2.*x*系，3.*x*系のそれぞれの中では，後方互換性はよく保たれているので，通常いずれかの最新版をインストールすればおそらく問題ありません．

● その9：Linux/Windows/Mac…どのコンピュータでも動く

Pythonの標準の実装(CPython)は，Windows，Mac，Linuxをはじめ，多くの環境で動作しています．ラズベリー・パイについても，Linuxのディストリビューションである Raspbianに標準で含まれています．iOS/Android上のアプリとして動くものもあり，筆者は電卓代わりに用いています．

またCPythonとは別に，ほぼ互換の言語として，Jython，IronPythonというものがあります．これらについては，Pythonのライブラリの一部が使えなくなる代わりに，それぞれJavaクラス・ライブラリ，.NET Frameworkの機能をフルに使うことができます．自作のコードとの連携も問題なくできますから，もしすでにJavaやC#，VB.netなどで開発しているなら，それらの言語で書かれたコード資産を生かしつつPythonで開発することもできます．

● その10：他の言語との連携も行いやすい

PythonとPythonのライブラリの組み合わせは，それだけでも強力ではありますが，他の言語にもそれぞれ有用な既存コードやライブラリがありますから，往々にして他言語の力に頼りたいことがあることも事実です．例えば，高度な最適化やシミュレーションを行いたい場合には，MATLABで記述された既存のコードを活用したい場合もあるでしょうし，統計処理においては，R言語で記述されたアルゴリズムを実行したいこともあるでしょう．

あるいは，大規模な処理を行う際，高速化のために一部の処理をC言語で書きたいこともあるかもしれません．そのような場合にも，Pythonを中心軸とし，一部の機能を他の言語と連携させることで，Pythonの機能と他の言語の機能の「いいとこどり」をすることができます．

もともとPythonをはじめとするスクリプト言語の利点の一つとして，他の環境との連携を行いやすいことが挙げられます．一部の処理を他の言語で行いたい場合は，

(1) Pythonのデータを他の環境に流し込む
(2) 他の環境で計算させる
(3) 得られた結果をPythonのデータとして格納する

ことが必要ですが，これらのコードを比較的容易に書くことができるのです．

特にMATLABやRをはじめ，数多くの言語との間には，データと処理を連携させるためのライブラリがすでにありますから，たいていの場合，ライブラリを用いるだけで目的を達成できるでしょう．

また，PythonにはC/C++言語で機能拡張するための仕組みが備わっており，時間のかかる関数のみをC言語のモジュールに置き換えることができます．

● その11：C/C++で書かれたプログラム中にPythonを組み込むことができる

普段C/C++で開発しているが，Pythonの柔軟さやライブラリの機能を取り入れたいという人のために，逆にPythonにはC/C++の組み込み言語として使う手段が用意されています．Pythonの関数を呼び出して使うことができることはもちろん，逆に組み込まれたPythonがプログラム本体の関数を使うことさえできます．

さとう・りょうへい

第2章 これからのコンピュータの当たり前！

IoT/データ解析にピッタリ！ おすすめPythonライブラリ事典101

世界の8万個ものライブラリから整理してみた

斉藤 直希

表1 紹介するライブラリあれこれ

表番号	項目	詳細
2	①科学/データ解析	数値計算，信号処理，統計処理など
3	②学習/認識	機械学習，深層学習，ニューラル・ネットワークなど
4	③画面制御	GTK+などのGUI，OpenGLなどのグラフィック・ライブラリ
5	④マルチメディア	音声，画像，動画など
6	⑤ウェブ開発	ウェブ・フレームワーク，HTMLパーサなど
7	⑥ネットワーク/通信	各種プロトコル実装(SSH, SMB)，パケット解析，SNSなど
8	⑦データベース	PostgreSQL，MySQLなどのデータベース・ドライバ
9	⑧ドライバ	GPIO，シリアル，ラズベリー・パイ，ロボットなど
10	⑨自然言語処理	形態素解析などの自然言語処理

「ライブラリ」とは一般に「多くのアプリケーションで利用することを目的としてプログラムをまとめたもの」です．よく使われる機能がライブラリとして提供されていれば，それを利用することで目的のプログラムをより短期間に作成できます．

Pythonは充実した標準ライブラリ注をもち，テキスト処理，数値処理，ファイル処理，マルチメディア処理，ネットワーク処理などのさまざまな処理に使用することができます(1)．

標準ライブラリ以外にも，科学計算やウェブ開発など種々の用途で利用可能な数多くのライブラリがあります．

Pythonのパッケージを探す場合，PyPI(Python Package Index)と呼ばれるPythonパッケージのリポジトリ・サイトが便利です(2)．また，Python Software Foundation(PSF)が運営するwikiには定番のライブラリを紹介するページもあり，ライブラリを探す場合の参考になります(3)．

Pythonのライブラリは非常に多くのものがあります(**表1**)．その(標準ライブラリを除く)一部の例を**表2**～**表10**に示します．科学計算やネットワーク処理，自然言語処理に至るまで，さまざまなライブラリが公開されています．

注：Pythonでは「ライブラリ」に相当する用語として「モジュール」および「パッケージ」も用いられるが，ここでは主に「ライブラリ」を使用する．詳しくは文献(4)を参照．

①科学/データ解析

Pythonは科学計算や数値解析などのライブラリが充実しています(**表2**)．Pythonは科学技術計算の用途でよく利用されています．

NumPyというベクトル/行列演算を扱う高速な数値演算ライブラリがPython言語公開後の比較的早い段階(1995年)から公開され，広く利用されています．調査(5)でも利用者の割合が高いことが分かります．NumPyをベースにしたライブラリも多く存在します．例えばSciPyはNumPyをベースに科学技術計算を追加するライブラリです．matplotlibはNumPyにグラフ描画機能を追加します．

SymPyは記号計算を処理するためのライブラリです．数学ソフトウェア・システムSageMathの一部として使用されています．

SageMathはPythonによるオープンソースの数学システムで，MATLABやMathematicaなどの代替を目指すソフトウェアとして利用されています．

Theano はNumPy同様の数値計算ライブラリです．

GPUを使用した高速な計算も行うことが可能です．

pandasはデータ解析を支援するライブラリです．時系列データの操作を扱うのに便利です．

NetworkXはノードとエッジからなる数学的なグラフを扱うためのライブラリです．

物理的なシミュレーションを行うためのライブラリもあり，例えば無線シミュレーションのためのGNU Radio，scikit-rfがあります．

数値解析用途で使われる他の言語をPythonから利用するためのライブラリもあります．Oct2PyはGNU octaveを，RPy2は統計処理やデータ解析で知られているR言語を利用するためのライブラリです．

表2　科学/データ解析ライブラリ

番号	名　称	説　明	pipパッケージ名注1	プロジェクトURL	ソースコード・リポジトリURL（プロジェクトとは別に存在する場合）	ライセンス（表11参照）
1	GNU Radio	ソフトウェア・ラジオ（Software Defied Radio，SDR）のためのツール・キット		http://gnuradio.org/		GNU GPLv3
2	matplotlib	NumPyのためのグラフ描画ライブラリ．さまざまな種類のグラフを描画できる．基本的に2Dグラフの描画を行うが，3Dグラフのプロット機能も追加されつつある	matplotlib	http://matplotlib.org/	https://github.com/matplotlib/matplotlib/	Python Software Foundation License
3	NetworkX	数学的なグラフを扱うためのライブラリ	networkx	https://networkx.github.io/	https://github.com/networkx/networkx	修正BSDライセンス
4	NumPy	数値計算を効率的に行うための拡張モジュール．型付の多次元配列の定義やそれを操作するためのライブラリを提供する	numpy	http://www.numpy.org/	https://github.com/numpy/numpy	修正BSDライセンス
5	Oct2Py	PythonからOctaveのMファイルや関数を呼び出すためのライブラリ	oct2py	http://blink1073.github.io/oct2py/	https://github.com/blink1073/oct2py	MITライセンス
6	pandas	データ構造およびデータ解析用ツール．数表や時系列データを操作するためのデータ構造と操作を提供する	pandas	http://pandas.pydata.org/	https://github.com/pydata/pandas	修正BSDライセンス
7	RPy2	R言語へのPythonインターフェース	rpy2	http://rpy2.bitbucket.org/		GNU GPLv2またはそれ以降
8	Sage Math	数学ソフトウェア・システム．NumPy，SciPyなどの既存の数学関連ソフトウェアを同じインターフェースで使えるようにしたもの．ラズパイ向けにSageMathを再コンパイルしたもの		http://www.sagemath.org/	https://github.com/sagemath/sage ラズパイ向け https://github.com/ArchimedesPi/SageMathematics-raspi	GNU GPLv3
9	scikit-rf	RF/マイクロ波工学のためのツール．ミリ波などを対象にした回路網のシミュレーションやグラフ作成	scikit-rf	http://scikit-rf-web.readthedocs.io/	https://github.com/scikit-rf/scikit-rf	修正BSDライセンス
10	SciPy	数学，科学，工学などのライブラリ．例えば線形代数演算，FFT，最適化，統計処理，補完，積分，信号解析，画像処理などの機能を提供する	scipy	https://www.scipy.org/		修正BSDライセンス
11	SimPy	離散型シミュレーション・ライブラリ	simpy	https://simpy.readthedocs.io/	https://bitbucket.org/simpy/simpy	MIT ライセンス
12	SymPy	代数計算ライブラリ．Pythonで記号計算ができる．例えば四則演算に加え式の簡約化や展開，記号置換などの処理ができる	sympy	http://www.sympy.org/en/index.html	https://github.com/sympy/sympy	修正BSDライセンス
13	Tensor Flow	データ・フロー・グラフを用いた数値計算フレームワーク．ラズパイ向けの非公式のパッケージが公開されている		https://www.tensorflow.org/	https://github.com/tensorflow/tensorflow ラズパイ向け https://github.com/samjabrahams/tensorflow-on-raspberry-pi	Apache License 2.0
14	Theano	数値計算ライブラリ．行列演算のための関数を提供する．NumPy/SciPyの代わりとして利用できる	theano	http://www.deeplearning.net/software/theano/	https://github.com/Theano/Theano	修正BSDライセンス

注1：pip 8.1.2で確認．pipについてはコラム1参照

②学習/認識

Pythonは科学計算用のライブラリが充実していますが，それをベースとした機械学習やパターン認識などのライブラリも各種提供されています(表3).

例えば深層学習や機械学習ライブラリとしてneon, Chainer, Keras, scikit-learn, SHOGUN, PyBrainなどや，それに関連した機能を提供するライブラリがあります.

ほかにはクラウド・サービスとして提供されている認識や学習などの機能をPythonから利用するためのライブラリも存在します. 例えばGoogle Cloud Vision API向けのライブラリやClarifai APIを利用するためのライブラリがあります.

表3　学習/認識ライブラリ

番号	名 称	説 明	pipパッケージ名	プロジェクトURL	ソースコード・リポジトリURL(プロジェクトとは別に存在する場合)	ライセンス(表11参照)
1	Chainer	深層学習フレームワーク	chainer	http://chainer.org/	https://github.com/pfnet/chainer	MITライセンス
2	clarifai-python	Clarifai APIのPythonクライアント	clarifai	https://www.clarifai.com/	https://github.com/Clarifai/clarifai-python	Apache License 2.0
3	Google Cloud Vision API	画像認識/分類用API	visionary	https://cloud.google.com/vision/	https://github.com/GoogleCloudPlatform/cloud-vision	Apache License 2.0
4	Keras	深層学習ライブラリ	keras	http://keras.io/	https://github.com/fchollet/keras	MITライセンス
5	Lasagne	ニューラル・ネットワークの構築および学習のための軽量ライブラリ	lasagne	http://lasagne.readthedocs.io/en/latest/	https://github.com/Lasagne/Lasagne/	MITライセンス
6	LIBLINEAR	線形分類器ライブラリ		https://www.csie.ntu.edu.tw/~cjlin/liblinear/	https://github.com/cjlin1/liblinear	修正BSDライセンス
7	LIBSVM	サポート・ベクタ・マシン(SVM)ライブラリ		https://www.csie.ntu.edu.tw/~cjlin/libsvm/	https://github.com/cjlin1/libsvm	修正BSDライセンス
8	neon	深層学習フレームワーク	nervananeon	http://neon.nervanasys.com/	https://github.com/NervanaSystems/neon	Apache License 2.0
9	Pattern	ウェブ・マイニング・ライブラリ	pattern	http://www.clips.ua.ac.be/pattern	https://github.com/clips/pattern	修正BSDライセンス
10	PyBrain	機械学習ライブラリ	pybrain, pybrain2	https://github.com/pybrain/pybrain/wiki/installation	https://github.com/pybrain/pybrain	修正BSDライセンス
11	scikit-learn	機械学習ライブラリ	scikit-learn	http://scikit-learn.org/stable/	https://github.com/scikit-learn/scikit-learn	修正BSDライセンス
12	SHOGUN	機械学習ライブラリ	shogun	http://www.shogun-toolbox.org/	https://github.com/shogun-toolbox/shogun	GNU LGPLv3またはそれ以降(一部例外を除く)
13	Statsmodels	データ探索，統計モデルの推定，統計手法を実施するためのPythonモジュール	statsmodels	http://statsmodels.sourceforge.net/	https://github.com/statsmodels/statsmodels/	修正BSDライセンス
14	visionary	Google Cloud Vision APIのためのクライアント・ライブラリ	visionary	https://github.com/shafaua/visionary		Apache License 2.0

③画面制御

解析したデータを理解するためにはそれを映像や音声などの形で表現するのが効果的です．そして，利用者が表示内容の切り替えなどの指示を出すにはGUIなどにより画面を操作できる必要があります．このような画面の表示や操作をPythonから行うためのライブラリが公開されています（**表4**）．

表4　画面制御ライブラリ

番号	名　称	説　明	pipパッケージ名	プロジェクトURL	ソースコード・リポジトリURL（プロジェクトとは別に存在する場合）	ライセンス（表11参照）
1	Kivy	マルチタッチ・ユーザ・インターフェース・アプリのためのUIフレームワーク	kivy	https://kivy.org/	https://github.com/kivy/kivy	MITライセンス（一部例外を除く）
2	PyGObject（PyGI）	GObject Introspectionというライブラリを通してGNOMEプラットフォームを利用するためのライブラリ．GUIフレームワークGTKもこのライブラリを通して利用可能	PyGObject	https://wiki.gnome.org/Projects/PyGObject	https://git.gnome.org/browse/pygobject	GNU LGPLv2.1
3	PyOpenGL	OpenGLおよび関連APIへのクロス・プラットフォームなPythonバインディング	pyopengl	http://pyopengl.sourceforge.net/	https://code.launchpad.net/pyopengl	修正BSDライセンス，MITライセンス，など（ファイルにより異なる）
4	PyQt	GUIライブラリQtをPythonで利用できるようにしたもの．同様のライブラリとしてPySideがある	PyQt4, PyQt5	https://riverbankcomputing.com/software/pyqt/intro		GNU GPLv3または商用ライセンス
5	PySDL2	マルチメディア・ライブラリSDL2のラッパ・ライブラリ	pysdl2	http://pysdl2.readthedocs.io/en/latest/index.html	https://bitbucket.org/marcusva/py-sdl2	パブリック・ドメイン
6	wxPython	ウィジェット・ツール・キットwxWigetsのPythonラッパ		https://www.wxpython.org/	https://github.com/wxWidgets/wxPython	wxWindows Library Licence

コラム1　ライブラリの依存関係が解決できるパッケージ管理ソフトpip　斉藤 直希

ライブラリを利用するためには当然ながらそれがインストールされている必要があります．標準ライブラリはすぐに利用可能ですが，それ以外のライブラリを利用するにはインストールが必要です．

Pythonには独自に作成したライブラリを配布用にまとめたり，配布用のライブラリをインストールしたりするための機能が用意されています(14)(15)．しかし通常はパッケージ管理用のツールを利用してインストールを行うのが一般的です．そのようなツールはライブラリを適切な場所からダウンロードしたり，ライブラリ間の依存関係を解決したりする機能を持つため便利です．公式ドキュメントにはツールやライブラリをインストールする方法が紹介されています(16)(17)．代表的なツールはpip(18)で，大抵のライブラリはこれでインストールできます．

ただし，Pythonにはバージョンの問題があります．Pythonには互換性が保証されていない2系および3系のバージョンが存在し，ライブラリが特定のバージョンのみを対象とする場合があります．そのためPythonには仮想環境(19)を作成するための仕組みが用意されています．仮想環境を作成し，その環境でライブラリをインストールすることで，ライブラリに応じてPythonのバージョンを切り替えることができます．

仮想環境を扱うためのツールとしてPythonバージョン3.3から追加されたpyvenv(20)があります．しかし，それ以前のバージョンも扱う場合はpyenv(21)やAnaconda(22)に含まれるcondaが便利です．これにより，さまざまなPythonのバージョンをディレクトリごとに切り替えることができます．

④マルチメディア

音声，画像，動画などのマルチメディア・データをPythonで扱うためのライブラリもあります．データを動画像などの形で表現しようとすればメディア・データの表示処理が必要となります(**表5**)．

画像処理のためのライブラリとしてはPillow，OpenCV，scikit-imageなどがあります．少し特殊な用途として医療用画像を処理するためのライブラリもあります．ニューロ・イメージング・データを解析するためのNiPyやfMRIデータを処理するためのNeurosynthというライブラリがあります．

音声に関して，python-mpd2はミュージック・プレーヤ・デーモン(MPD)を通じて音声再生を行うためのクライアント・ライブラリをPythonで利用するためのライブラリです．音声合成のためのpython-espeakというライブラリもあります．

GSteramerは音声や動画などさまざまな種類のデータを利用したストリーミング・メディアを構築するためのフレームワークで，Pythonからも利用できます．

表5　マルチメディア・ライブラリ

番号	名　称	説　明	pipパッケージ名	プロジェクトURL	ソースコード・リポジトリURL(プロジェクトとは別に存在する場合)	ライセンス(**表11**参照)
1	GSteramer Python Binding	マルチメディア・フレームワーク．メディアを扱うコンポーネントのグラフを構築するためのライブラリ		https://gstreamer.freedesktop.org/modules/gst-python.html	https://cgit.freedesktop.org/gstreamer/gst-python	GNU LGPLv2.1
2	Neurosynth	fMRI(functional magnetic resonance imaging)データの自動合成のためのプラットフォーム．幾つかのツールから構成される	neurosynth	http://neurosynth.org/	https://github.com/neurosynth/neurosynth https://github.com/neurosynth/ace https://github.com/neurosynth/neurosynth-data	MIT ライセンス
3	NiPy	ニューロ・イメージング・データを解析するためのツール群	nipy	http://nipy.org/	https://github.com/nipy	修正BSDライセンス，Apache License 2.0(ツールにより異なる)
4	OpenCV	画像処理ライブラリ		http://opencv.org/	https://github.com/Itseez/opencv	修正BSDライセンス
5	Pillow	画像処理ライブラリ．PIL(Python Imaging Library)の後継	pillow	http://python-pillow.org/	https://github.com/python-pillow/Pillow	Python Imaging Library license
6	PyAudio	オーディオ・ライブラリPortAudioをPythonで利用するためのライブラリ．音声の再生や録音が可能	pyaudio	https://people.csail.mit.edu/hubert/pyaudio/	http://people.csail.mit.edu/hubert/git/?p=pyaudio.git	MIT ライセンス
7	python-espeak	音声合成プログラムespeakをPythonから利用するためのライブラリ		https://launchpad.net/python-espeak	http://bazaar.launchpad.net/~rainct/python-espeak/trunk/files	GNU GPLv3
8	python-mpd2	ミュージック・プレイヤ・デーモン(MPD)のためのクライアント・インターフェースを提供するためのライブラリ	python-mpd2	https://github.com/Mic92/python-mpd2		GNU LGPLv3
9	scikit-image	画像処理ライブラリ(Scipyのツール・キット)	scikit-image	http://scikit-image.org/	https://github.com/scikit-image/scikit-image/	修正BSDライセンス

⑤ウェブ開発

Pythonはウェブ・アプリケーション開発にもよく利用されています．各種のウェブ・フレームワークがPython向けに提供されています（**表6**）．中でもDjango, Flask, Pyramidは比較的利用者が多いウェブ・フレームワークです[5][6]．それ以外にもBottleという軽量なウェブ・フレームワークもあります．

フレームワーク以外では，HTTPパケットやAtom/RSSといったパケットの解析を行うためのライブラリもあります．

表6　ウェブ開発ライブラリ

番号	名　称	説　明	pipパッケージ名	プロジェクトURL	ソースコード・リポジトリURL（プロジェクトとは別に存在する場合）	ライセンス（表11参照）
1	Beautiful Soup	スクレイピングのためのHTMLパーサ・ライブラリ	beautifulsoup beautifulsoup4	https://www.crummy.com/software/BeautifulSoup/		MITライセンス
2	Bottle	ウェブ・アプリ開発のためのマイクロ・フレームワーク	bottle	http://bottlepy.org/docs/dev/index.html	https://github.com/bottlepy/bottle	MITライセンス
3	Cheetah	テンプレート・エンジン	cheetah	http://www.cheetahtemplate.org/	https://github.com/cheetahtemplate/cheetah	MITライセンス
4	Django	ウェブ・フレームワーク	django	https://www.djangoproject.com/	https://github.com/django/django	修正BSDライセンス
5	feed parser	RSS/Atomパーサ・ライブラリ	feedparser		https://github.com/kurtmckee/feedparser	2条項BSDライセンス
6	Flask	Werkzeugベースのマイクロ・フレームワーク	flask	http://flask.pocoo.org/	https://github.com/pallets/flask	修正BSDライセンス
7	Plone	コンテンツ管理システム	plone	https://plone.com/	https://www.openhub.net/p/plone	GNU GPLv2
8	Pyramid	ウェブ・アプリケーション・フレームワーク	pyramid	http://www.pylonsproject.org/projects/pyramid/about	https://github.com/Pylons/pyramid	Repoze Public License
9	Tornado	Pythonウェブ・フレームワークおよび非同期ネットワーキング・ライブラリ	tornado	http://www.tornadoweb.org/	https://github.com/tornadoweb/tornado	Apache License 2.0
10	Twisted	イベント駆動型のネットワーク・プログラミングのためのフレームワーク	twisted	https://twistedmatrix.com/trac/	https://github.com/twisted/twisted	MITライセンス

コラム2　世界の定番プログラミング言語Python

斉藤 直希

Pythonとは，1991年にオランダのグイド・ヴァン・ロッサム（Guido van Rossum）によって開発された汎用のプログラミング言語です．プログラムはインタプリタ上で動作し，Windows，Mac，UNIX系OSなどさまざまな実行環境で動作可能です．

Pythonはシンプルで柔軟性が高いと言われています．主に海外での人気が高く，プログラミング言語の人気ランキングにおいて上位にランクされることが多い言語です[7][8][9][10][11]．また2014年の調査ですが，全米のコンピュータ・サイエンス・コースを持つ大学の39校に対する調査で，初心者向けプログラミング教育教材としてPythonが最もカリキュラムに取り入れられているとの報告もあります[12]．そのぶん利用者の幅が広い言語といえるかもしれません．

国内ではPythonは海外に比べマイナな存在でしたが，近年では人工知能，機械学習，ロボットなどの応用分野を中心に注目されています．ビズリーチの運営する検索サイト「スタンバイ」の調査では，Pythonの技術者の平均年収が最も高いとの報告もあります[13]．

⑥ネットワーク/通信

さまざまなネットワーク・プロトコルを実装したライブラリが提供されています(**表7**).

例えばHTTP, SSH, SMBなどのほか, HTTP/2, WebSocket, MQTT, SIPなどのプロトコルも利用できます.

Wi-FiやBluetoothの無線通信を管理するためのライブラリもあります. プロトコル以外では, パケットのキャプチャや解析を行うためのライブラリもあります.

表7 ネットワーク/通信ライブラリ

番号	名 称	説 明	pipパッケージ名	プロジェクトURL	ソースコード・リポジトリURL(プロジェクトとは別に存在する場合)	ライセンス(**表11**参照)
1	Braintree	決済ゲートウェイBraintreeのPythonライブラリ	braintree	https://developers.braintreepayments.com/start/hello-server/python	https://github.com/braintree/braintree_python	MITライセンス
2	Hyper-h2	HTTP/2プロトコル・スタック	h2	http://python-hyper.org/h2/	https://github.com/python-hyper/hyper-h2/	MITライセンス
3	knx	小規模KNX/EIBライブラリ	knx	https://github.com/mfussenegger/knx		MITライセンス
4	Mosquitto	MQTTバージョン3.1 Pythonクライアント・ライブラリ		http://mosquitto.org/	https://github.com/eclipse/mosquitto	EPLおよびEDL
5	NCAP	ネットワーク・キャプチャ・ライブラリ		https://www.dns-oarc.net/tools/ncap		修正BSDライセンス
6	Paho	Paho(MQTTクライアント・ライブラリ)のPython実装	paho-mqtt	http://www.eclipse.org/paho/clients/python/	https://github.com/eclipse/paho.mqtt.python	EPLおよびEDL
7	Paramiko	SSHv2ライブラリ	paramiko	http://www.paramiko.org/	https://github.com/paramiko/paramiko	GNU LGPLv2.1
8	pjsua	マルチメディア通信ライブラリPJSIPのクライアント・ライブラリ	pjsua	http://www.pjsip.org/python/pjsua.htm	https://trac.pjsip.org/repos/wiki/Python_SIP_Tutorial	GNU GPLv2またはそれ以降
9	PyBluez	Bluetooth用のPython拡張モジュール	pybluez	http://karulis.github.io/pybluez/	https://github.com/karulis/pybluez	GNU GPLv2またはそれ以降
10	pyClamd	アンチウィルス・エンジンClamAVインターフェース	pyclamd	http://xael.org/pages/pyclamd-en.html	https://bitbucket.org/xael/pyclamd	GNU LGPLv3またはそれ以降
11	pyknx	Linknx(KNXによる自動制御プラットフォーム)のPythonバインディング	pyknx	https://pypi.python.org/pypi/pyknx	https://github.com/2franix/pyknx	GNU GPLv3
12	pysmb	クライアント・サイドSMB/CIFSプロトコルのPython実装(SMB1 and SMB2)	pysmb	https://miketeo.net/wp/index.php/projects/pysmb	https://github.com/miketeo/pysmb	zlibライセンス
13	pysmbc	SAMBAクライアント・ライブラリのPythonバインディング	pysmbc	http://cyberelk.net/tim/software/pysmbc/	https://git.fedorahosted.org/cgit/pysmbc.git/	GNU GPLv2
14	python-nmap	nmapポート・スキャナ用ライブラリ	python-nmap	http://xael.org/pages/python-nmap-en.html	https://bitbucket.org/xael/python-nmap	GNU GPLv3
15	Requests	HTTPライブラリ. 標準ライブラリのurllibよりも簡単にHTTP通信処理を記述できる	requests	http://docs.python-requests.org/en/master/	https://github.com/kennethreitz/requests	Apache License 2.0
16	Scapy	パケット操作のためのライブラリおよび対話型プログラム	scapy	http://www.secdev.org/projects/scapy/	https://github.com/secdev/scapy/	GNU GPLv2

表7　ネットワーク/通信ライブラリ（つづき）

番号	名　称	説　明	pipパッケージ名	プロジェクトURL	ソースコード・リポジトリURL（プロジェクトとは別に存在する場合）	ライセンス（表11参照）
17	slacker	コミュニケーション・ツールSlackのAPI	slacker		https://github.com/os/slacker	Apache License 2.0
18	Tweepy	Twitter APIアクセス用ライブラリ	tweepy	http://www.tweepy.org/	https://github.com/tweepy/tweepy	MITライセンス
19	websocker-client	WebSocketクライアント・ライブラリ	websocket-client	https://github.com/liris/websocket-client		GNU LGPLv2.1
20	wifi	Wi-Fiネットワークの検出，設定，接続用ライブラリ	wifi	https://pypi.python.org/pypi/wifi	https://github.com/rockymeza/wifi	2条項BSDライセンス
21	ws4py	WebSocketライブラリ	ws4py	https://ws4py.readthedocs.io/	https://github.com/Lawouach/WebSocket-for-Python	修正BSDライセンス

⑦データベース

科学計算，機械学習，画像認識など，大量のデータを扱う用途でPythonが使われるようになるとデータベース処理が必要となります．PythonにはPostgreSQL，MySQLなどの既存のデータベース・システムを利用可能にするためのライブラリが公開されています（**表8**）．

まず，PostgreSQLをPythonから利用するためのライブラリとしてpsycopg，pg8000，py-postgresqlなどがあります．MySQL はPython用のライブラリが含まれているためそのまま利用できます．その他のデータベースとしてはMongoDB，Apache Cassandra，Oracle DB, Oracle Berkeley DB用のライブラリもあります．

表8　データベース・ライブラリ

番号	名　称	説　明	pipパッケージ名	プロジェクトURL	ソースコード・リポジトリURL（プロジェクトとは別に存在する場合）	ライセンス（表11参照）
1	bsddb3	Oracle Berkeley DBのPythonインターフェース	bsddb3	https://www.jcea.es/programacion/pybsddb.htm	http://hg.jcea.es/pybsddb/	修正BSDライセンス
2	cx_Oracle	OracleデータベースへのPythonインターフェース	cx_oracle	http://cx-oracle.sourceforge.net/	https://bitbucket.org/anthony_tuininga/cx_oracle	Python Software Foundation License
3	MySQL Connector/Python	MySQLドライバ	mysql-connector	http://dev.mysql.com/doc/connector-python/en/		GNU GPL v2 または商用ライセンス
4	pg8000	PostgreSQLドライバ	pg8000	http://pythonhosted.org/pg8000/	https://github.com/mfenniak/pg8000	修正BSDライセンス
5	Psycopg	PostgreSQLアダプタ	psycopg	http://initd.org/psycopg/	https://github.com/psycopg/psycopg2	GNU LGPLv3またはそれ以降（一部例外あり），またはZPL
6	PyMongo	PythonでMongoDBを利用するためのモジュール	pymongo	https://api.mongodb.com/python/current/	https://github.com/mongodb/mongo-python-driver/	Apache License 2.0
7	py-postgresql	PostgreSQL向けドライバおよびツール	py-postgresql	http://python.projects.pgfoundry.org/	https://github.com/python-postgres/fe	修正BSDライセンス
8	Python Cassandra Driver	Apache CassandraのためのPythonドライバ	pycassa	http://datastax.github.io/python-driver/index.html	https://github.com/datastax/python-driver	Apache License 2.0

⑧コンピュータ・ボード向けのドライバ

最近ではラズベリー・パイなど，Pythonの動作可能な小型コンピュータが手軽に利用できるようになりました．そのような小型コンピュータは，センサ・ネットワークのノードのようにセンサを接続してデータ収集し，ネットワークへデータを送出するというような使い方が簡単にでき，活用の幅が広がります．

Pythonにはシリアル・ポートなどハードウェアを制御するためのライブラリがあります．ラズベリー・パイにはGPIOを利用するためのライブラリもあります．それらのライブラリ(**表9**)を使うと，一つのスクリプトでデータ処理やネットワーク処理などの既存ライブラリとまとめて扱うことができ便利です．

表9　コンピュータ・ボード向けのドライバ

番号	名　称	説　明	pipパッケージ名	プロジェクトURL	ソースコード・リポジトリURL(プロジェクトとは別に存在する場合)	ライセンス(表11参照)
1	DroneKit-Python	MAVLinkプロトコルを用いてドローンと通信を行うためのライブラリ	dronekit	http://python.dronekit.io/	https://github.com/dronekit/dronekit-python	Apache License 2.0
2	Gertbot	ラズベリー・パイ向けモータ・ドライバ基板Gertbot用ドライバ		http://www.gertbot.com/		不明(明記なし)
3	GoPiGo	ラズベリー・パイ向けロボット・キットGoPiGoを制御するためのドライバ	gopigo	http://www.dexterindustries.com/GoPiGo/	https://github.com/DexterInd/GoPiGo	GNU GPLv3
4	Navio	ラズベリー・パイ向けオートパイロット・シールド基板Navioボード用ドライバ.		https://github.com/emlid/Navio		修正BSDライセンス
5	nfcpy	NFC向けPythonモジュール		http://nfcpy.readthedocs.io/	https://github.com/nfcpy/nfcpy	EUPL v1.1またはそれ以降
6	Picamera	ラズベリー・パイのカメラ・モジュールを扱うためのライブラリ	picamera	http://picamera.readthedocs.io/	https://github.com/waveform80/picamera	修正BSDライセンス
7	pylibftdi	FTDIデバイス・アクセスのためのPythonインターフェース	pylibftdi	https://pylibftdi.readthedocs.io/	https://bitbucket.org/codedstructure/pylibftdi	MIT ライセンス
8	PyParallel	パラレル・ポートへのアクセスをカプセル化したモジュール	pyparallel	https://github.com/pyserial/pyparallel		修正BSDライセンス
9	pySerial	シリアル・ポート・アクセス・ライブラリ	pyserial	https://github.com/pyserial/pyserial		修正BSDライセンス
10	python-lirc	赤外線リモコン・ライブラリLIRCのPython拡張	python-lirc	https://github.com/tompreston/python-lirc		GNU GPLv3
11	python-smbus	SMBusライブラリのPythonバインディング		http://www.lm-sensors.org ※原稿執筆時点ではアクセス不可.	https://github.com/groeck/i2c-tools ※python-smbusパッケージに対し修正が加えられている.	GNU GPLv2以降，GNU LGPL v2.1以降(ツールによる)
12	PyVideo Core	ラズベリー・パイでGPGPUを行うためのライブラリ		https://github.com/nineties/py-videocore		MIT ライセンス
13	RPi.GPIO	ラズベリー・パイ向けGPIO制御用Python モジュール	rpi.gpio	https://sourceforge.net/projects/raspberry-gpio-python/		MIT ライセンス
14	smbus2	python-smbusパッケージ互換のPythonライブラリ	smbus2	https://pypi.python.org/pypi/smbus2/0.1.2	https://github.com/kplindegaard/smbus2	MIT ライセンス
15	WebIOPi	ラズベリー・パイ向けIoTフレームワーク		http://webiopi.trouch.com/	https://sourceforge.net/projects/webiopi/	Apache License 2.0
16	WiringPi	ラズベリー・パイ向けGPIOインターフェース・ライブラリ	wiringpi, wiringpi2	http://wiringpi.com/		GNU LGPLv3

⑨自然言語処理

SNSの分析やアンケートの分析など，アプリケーション開発において自然言語（例えば日本語）を解析したい場合があります．そのような解析処理がPythonから利用できると，その後のデータ処理にスムーズに移行できて便利です（**表10**）．

自然言語処理のためのライブラリとしてはNLTKや形態素解析のためのmecab，Janomeといったライブラリがpythonから利用できます．

＊　　＊　　＊

各ライブラリにはライセンスがあります（**表11**）．使用前には中身をよく確認してください．

表10　自然言語処理ライブラリ

番号	名　称	説　明	pipパッケージ名	プロジェクトURL	ソースコード・リポジトリURL（プロジェクトとは別に存在する場合）	ライセンス（表11参照）
1	Janome	Pythonで記述された辞書内包の形態素解析ライブラリ	janome	http://mocobeta.github.io/janome/	https://github.com/mocobeta/janome	Apache License 2.0
2	mecab-python3	形態素解析エンジンMeCabに対するPythonラッパー・ライブラリ（Python3対応版）	mecab-python3	https://pypi.python.org/pypi/mecab-python3	https://github.com/SamuraiT/mecab-python3	GNU GPLv2，GNU LGPLv2.1，または修正BSDライセンス
3	NLTK	自然言語処理用のツール・キット	nltk	http://www.nltk.org/	https://github.com/nltk/nltk	Apache License 2.0

注1：pip 8.1.2で確認

表11　ライセンス

ライセンス名称	概　要	参　照
MITライセンス	マサチューセッツ工科大学を起源とするソフトウェア・ライセンス．無償で自由に使用，修正，再頒布などが可能．利用上の制約が非常に少ないライセンス．多くのソフトウェアに適用されている．2012年の調査ではGitHubで最も採用されたライセンス	https://opensource.org/licenses/mit-license.php
（2条項or3条項）BSDライセンス	カリフォルニア大学で策定されたライセンス．BSDライセンスには幾つかの種類があり，ライセンス内の条項数によって｛4条項，3条項，2条項｝BSDライセンスと区別される．現在では3条項（修正BSDライセンス）または2条項のライセンスが多く使われる． 自由に使用，修正，再頒布が可能．3条項ライセンスの場合，書面による許可なくライセンスに記載の組織名やコントリビュータの名前を派生した製品の宣伝や販促で使用禁止	（2条項） https://opensource.org/licenses/BSD-2-Clause （3条項） https://opensource.org/licenses/BSD-3-Clause
Apache License 2.0	Apacheソフトウェア財団によるライセンス．BSDライセンスに対してその不足分を補う形で作成されており，頒布の際は著作権，特許，商標，および改変部分について告知する必要があるなどの要求事項が増えている． 自由な使用，修正，再頒布が可能．また特許条項があり，ライセンサは頒布するソフトウェアに含まれる特許について無償でライセンス付与するように規定されている	http://www.apache.org/licenses/LICENSE-2.0
GNU General Public License（GPL）	フリー・ソフトウェア財団（FSF）により公開されているライセンス．複数のバージョンが存在する．最新版はバージョン3であるが，バージョン2も利用される．ソフトウェアを自由に使用，修正，頒布，改変を公開できる．コピーレフト（著作権を保持したまま2次著作物も含めて，全ての者が著作物を利用・再配布・改変できなければならないという考え方）に基づき，2次著作物に対してもGPLであることを要求する	（バージョン3） http://www.gnu.org/licenses/gpl-3.0.en.html （バージョン2） http://www.gnu.org/licenses/old-licenses/gpl-2.0.txt
GNU Lesser General Public License（LGPL）	FSFにより公開されているライセンス．複数のバージョンが存在する．最新版はバージョン3であるが，バージョン2.1も利用されている．GPLと同じ内容であるが，ライブラリへの適用を想定してコピーレフトの適用を若干緩めた内容となっている（準コピーレフト型）．再頒布する際のライセンスとしてLGPLを適用する必要があるが，組み合わせる対象の別ソフトウェアにまでLGPLを適用する必要はない	（バージョン3） http://www.gnu.org/licenses/lgpl-3.0.en.html （バージョン2.1） http://www.gnu.org/licenses/old-licenses/lgpl-2.1.en.html

表11　ライセンス（つづき）

ライセンス名称	概　要	参　照
zlib ライセンス	zlibおよびlibpngなどのライブラリの配布で使用されるライセンス．誰でも商用アプリを含む任意目的で自由にソフトウェアを利用，変更，頒布可能	http://zlib.net/zlib_license.html
Eclipse Public License (EPL) 1.0	Eclipse FoundationによりEclipseなどのために使われるライセンス．自由に利用，改変，頒布可能．準コピーレフト型のライセンスであり，再頒布する際のライセンスとしてEPLを適用する必要があるが，組み合わせられる別のソフトウェアまでEPLを適用する必要はない．また，特許が含まれている場合，ライセンサは無償でライセンス付与することと規定されており，無償で利用することができる	http://www.eclipse.org/legal/epl-v10.html
Eclipse Distribution License (EDL) 1.0	Eclipse Foundationより公開されているライセンス．内容はほぼ3条項BSDライセンスと同様	https://eclipse.org/org/documents/edl-v10.php
Python Software Foundation License	主にPythonプロジェクトの配布用に用いられるライセンス．BSDによく似たライセンスで自由に使用，変更，頒布が可能．コードをオープンにせず元のソースコードの変更を行うこともできる	https://www.python.org/psf/license/
Python Imaging Library license	Python Imaging Libraryで用いられるライセンス．無償で自由に使用，複製，修正，頒布が可能．書面による許可なくSecret Labs ABの名称や著者名を使用できない	http://www.pythonware.com/products/pil/license.htm
Zope Public License (ZPL) v2.1	主にZopeアプリケーション・サーバで用いられるライセンス．BSDライセンスに近いが，商標利用禁止や変更の文書化などの条項が追加されている	http://directory.fsf.org/wiki?title=License:ZopePLv2.1
wxWindows Library Licence	wxWidgetsで用いられるライセンスLGPLに幾つかの例外条項を付け加えたもの	http://www.wxwidgets.org/about/licence/
European Union Public Licence (EUPL)	EU内の政策執機関であるEuropean Commissionで電子政府サービスを開発する機関ECIDABCにより作成されたライセンス．コピーレフト型のライセンス．GPL v2をもとに欧州での利用を想定した形に変更したもの	https://joinup.ec.europa.eu/community/eupl/

◆参考文献◆

(1) Python標準ライブラリ（バージョンごとに存在，下記はバージョン3.5）．
http://docs.python.jp/3.5/library/
(2) PyPI - the Python Package Index.
https://pypi.python.org/pypi
(2)の後継サイトもある．PyPI - the Python Package Index-Warehouse (PyPI)．https://pypi.io/
(3) UsefulModules - Python Wiki.
https://wiki.python.org/moin/UsefulModules
(4) Python 3.5.1 ドキュメント「6. モジュール（module）」．
http://docs.python.jp/3.5/tutorial/modules.html
(5) Python Developers Survey 2016: Findings（JetBrains社による調査結果）．
https://www.jetbrains.com/pycharm/python-developers-survey-2016/
(6) PyCon JP 2015 アンケート（一般社団法人PyCon JP）．
https://pycon.jp/2015/ja/files/12/pyconjpquestionnaire0129.htm
(7) TIOBE Index for July 2016（TIOBE Software）．
http://www.tiobe.com/tiobe_index
(8) Interactive: The Top Programming Languages 2015 - IEEE Spectrum. http://spectrum.ieee.org/static/interactive-the-top-programming-languages-2015
(9) Language Trends on GitHub（GitHub）．
https://github.com/blog/2047-language-trends-on-github
(10) Stack Overflow Developer Survey 2016 Results - I. Most Popular Technologies.
http://stackoverflow.com/research/developer-survey-2016#technology
(11) What's the Best Programming Language to Learn in 2015-
https://www.sitepoint.com/whats-best-programming-language-learn-2015/
(12) Python is Now the Most Popular Introductory Teaching Language at Top U.S. Universities.
http://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-us-universities/fulltext
(13) プログラマー年収ランキング2016！（ビズリーチ）．
https://jp.stanby.com/media/programming_ranking/
(14) Python 3.5.1 ドキュメント「28.1. distutils-Python モジュールの構築とインストール」．
http://docs.python.jp/3.5/library/distutils.html
(15) Python Packaging User Guide.
https://packaging.python.org/
(16) Python 3.5.1 ドキュメント「Python モジュールのインストール」．
http://docs.python.jp/3.5/installing/index.html
(17) Tool Recommendations-Python Packaging User Guide documentation.
https://packaging.python.org/current/
(18) pip - pip 8.1.2 documentation.
https://pip.pypa.io/en/stable/
(19) PEP 405　Python Virtual Environments.
https://www.python.org/dev/peps/pep-0405/
(20) Python 3.5.1 ドキュメント「5. その他のツールとスクリプト」．
http://docs.python.jp/3.5/using/scripts.html
(21) pyenv. https://github.com/yyuu/pyenv
(22) Anaconda.
https://www.continuum.io/why-anaconda

さいとう・なおき

Appendix 1 apt-get/pip/import…基本ステップであっという間!

Pythonライブラリの基本的な使い方

三ツ木 祐介

Pythonのライブラリは数行から数十行のコードで動かせます．そのことを確かめるために，**表1**（次頁）に示すような画像系のライブラリの一部を動かしてみました．作業環境は以下です．

OS：Ubuntu 15.04（64ビット）
Pythonバージョン：2.7

なるべくPython 3にも対応しているものを挙げていますが，動作確認はPython 2.7で行っています．

表1（p.28）は以下の基準で選びました．

- 業界で「○○と言えばこのライブラリ」と有名なもの
- 画像処理，図形処理など画面周りの処理ができるもの
- GUIアプリケーションやゲームなどが作れると思うもの

● 画像処理/モーション解析/機械学習 OpenCV-Python

画像処理の定番であるOpenCVをPythonから使用するためのライブラリです．画像認識なども比較的簡単に実装できます．

▶インストール

```
$ sudo apt-get -y install python-opencv python-numpy
```

▶コード例（python_opencv.py）

```
import sys

import numpy as np
import cv2

img = cv2.imread(sys.argv[1], 0)
img = cv2.resize(img, (640, 480))
cv2.imshow('OpenCV sample', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

（注釈：OpenCVの使用に必要なモジュールのインポート／画像ファイルを読み込んでリサイズ／画像を表示）

▶実行例

```
$ python python_opencv.py testdata.jpg
```

コマンドの引き数に指定したJPEGファイルを読み込み，640×480にリサイズし，ウィンドウに表示します（**図1**）．実質5行で画像のロード/リサイズ/ウィンドウへの描画を行えます．



図1　OpenCV-Pythonの実行例

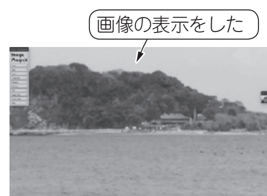


図2　Python Imaging Libraryの実行例

● 画像ファイルの読み込み/フィルタ/調整/保存 Python Imaging Library（PIL/Pillow）

画像取り込み，画像調整，表示を行うライブラリです．メンテナンスが停滞しているため，PILをFork（分岐）したPillowが使用されることが多いです．

▶インストール

```
$ sudo apt-get install -y python-pil
```

▶コード例（python_pil.py）

```
from PIL import Image
import sys

img = Image.open(sys.argv[1])
img.show()
```

（注釈：PILのモジュールをインポート／画像をロードしウィンドウに表示）

▶実行例

```
$ python python_pil.py testdata.jpg
```

図2に実行例を示します．実質2行で画像のロード，ウィンドウの描画を行うことができます．Image Magickのコンポーネントを使っているため，多機能である分，ウィンドウのカスタマイズなどは難しいかもしれません．

● GUIを提供する1：PyQt4

PyQt4は，Qt（キュート）の機能をPythonから呼び出すライブラリです．Qtは，アプリケーション・ユーザ・インターフェース・フレームワークを提供します．一度の開発で複数のプラットフォーム（Windows, Linux, Mac OS）をカバーできます．

▶インストール

```
$ sudo apt-get install -y python-qt4
```

表1 画像処理/表示やGUIのライブラリ

分 野	ライブラリ名	説 明	ライセンス	URL
画像処理	OpenCV-Python	Computer VisionのためのライブラリであるOpenCVをPythonから使用するためのライブラリ．OpenCVそのものはクロス・プラットフォームのライブラリであるため，Windows，MAC，Linuxと，さまざまなプラットフォームで動作する．基本的にはC++で使用されるために設計されているが，この豊富な機能をPythonでも使用できるようになっている画像処理/モーション解析/機械学習などができる	BSD	http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html
	Python Image Library (Pillow)	Linuxでの画像処理の定番であるImageMagickの機能をPythonから使用できる．Pythonで画像処理というと1番目か2番目に目に留まる．PILは開発が停滞しているためフォークであるPillowが使用されることが多い．画像ファイルの読み込み/フィルタ/調整/保存などができる	MIT-like	http://python-pillow.org/
GUI	Kivy	PythonでマルチタッチのUIを開発できるクロス・プラットフォームのライブラリ．少ない行数でかなりリッチなGUIが構築できる	MIT/LGPLv3	https://kivy.org
	PyQt	QtはLinuxデスクトップの代表格の一つであるKDEのベースとして使用されているライブラリ．Window，LinuxなどのPC以外にも組み込み用途などさまざまなプラットフォームでGUIを実装するために使われており幅広い使用実績がある．QtそのものはC++で使用されるために設計されているが，PyQt4はこれらの機能をPythonでも使用できるようにしたもの	GPLv3/商用	https://wiki.python.org/moin/PyQt4
	PyGTK	Adobe PhotoShopに対し，オープンソースで対抗しうるとの評価を得たGIMPを実装するために設計されたGTK+（The GIMP ToolKit）の機能をPythonから呼び出すライブラリ．GTK+はLinuxデスクトップにおいてKDEと対を成すGNOMEのベースとして使用されている．PyGTKはこれらの機能をPythonから使用できるようにしたもの	LGPLv3	http://www.pygtk.org/
	pyglet	オブジェクト指向のAPIを持つクロス・プラットホームのライブラリ．ゲームやリッチなGUIを持つアプリケーションの開発に使用される	BSD	https://bitbucket.org/pyglet/pyglet/wiki/Home
	wxPython	PC上でのGUIプログラミング黎明期からC++によるオブジェクト指向で設計され，Windows，Linux，Mac OS Xで動作する，クロス・プラットフォームのライブラリ．対応するプラットフォームの多さもさることながら，Python以外にもPerl，JavaScriptなどからその機能を使用できる	wxWidgets	http://www.wxpython.org/
図形処理	matplotlib	2D/3Dのグラフを描画する．単純なグラフだけではなくOpenCVなどで解析されたヒストグラムや音声データの波形などを簡単に可視化できる	matplotlib lic	http://matplotlib.org/mpl_toolkits/mplot3d/tutorial.html
	VPython	3Dオブジェクトの表示やアニメーションなどを簡単に行える．3行のプログラムで3Dオブジェクトが表示できる	Boost and MIT	http://vpython.org/
ゲーム・エンジン	cocos2d	iOSやAndroid向けのゲーム開発で有名なcocos2d-xのルーツとなるライブラリ．cocos2d-xは携帯向けの有名なゲームでも使用されている．cocos2d-xはC++での開発をターゲットにしているが，ルーツとなったcocos2dはpyglet上のフレームワークとして実装されており，Pythonでの開発をターゲットにしている．20行足らずのコードでアニメーションが実装できる	BSD	http://python.cocos2d.org/index.html
	Panda3D	3Dレンダリングおよびゲーム開発のためのフレームワーク．グラフィクスはもちろんオーディオやI/Oを含む．20行程度で3Dのシーンが読み込める	BSD-like	https://www.panda3d.org/
	PyGame	SDL（Simple DirectMedia Layer）のための Python 用ラッパ．2Dゲームや音楽，ビデオ再生などのマルチメディア・アプリケーションの開発に使用できる	LGPLv2.1	http://www.pygame.org/hifi.html
3Dエンジン	Soya3D	オブジェクト指向の「ハイレベルな」3Dエンジン．学習のしやすさと実行時のパフォーマンスが特徴	GPLv3	http://www.lesfleursdunormal.fr/static/informatique/soya3d/index_en.html
	Pi3D	ラズベリー・パイでの3Dプログラミング学習用に作成された，クロスプラットフォームのライブラリ．3Dプログラムを実行するには貧弱だった初期のラズベリー・パイでも，ある程度快適に動作させられるようにOpenGL ES 2.0をサポートしている．このためラズベリー・パイ以外にもOpenGL ES 2.0が実行できる他の組み込みプラットフォームでも，快適な動作が見込める．学習用途に用意されたAPIであるため，シンプルで分かりやすいAPIが特徴	MIT	http://pi3d.github.io/html/index.html
科学	PsychoPy	心理学実験環境を構築する．心理学実験で使われる定番の処理，画面に刺激を描いたり音声を鳴らしたり，反応時間を記録したりできる	GPLv3	http://www.psychopy.org/

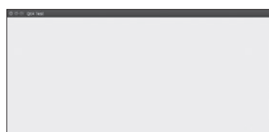

図3　PyQt4の実行例



図4　wxPythonの実行例

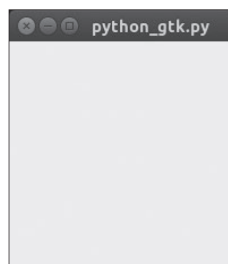


図5　PyGTKの実行例

▶コード例（python_qt4.py）

```
import sys
from PyQt4 import QtGui ←Qt4のモジュールをインポート

def main():                       ←メイン関数を定義
      app = QtGui.QApplication(sys.
                              argv)
                                  ウィンドウのタイトルをセットして表示
     w = QtGui.QWidget()
     w.setWindowTitle('Qt4 Test')
     w.show()

     sys.exit(app.exec_()) ←app.exec_()でメイン・ループを回している
if __name__ == '__main__':
     main()
```

▶実行例

```
$ python python_qt4.py
```

図3のように白い画面が表示されるだけです．このウィンドウは特に機能を持っていません．プログラミング次第でQt4のさまざまな機能を簡単に使用できます．

● GUIを提供する2：wxPython

wxPythonは，Pythonから利用できるGUIツール・キットです．中身はC++で書かれたwxWidgetsというライブラリをPythonのモジュールとしてラッピングしたものです．簡単にGUIアプリケーションを構築できます．

▶インストール

```
$ sudo apt-get install -y python-
wxgtk2.8
```

▶コード例（python_wx.py）

```
import wx ←wxのモジュールをインポート

app = wx.App()

f = wx.Frame(None, 1, 'wxPython
                         Test')  ←ウィンドウを表示
f.Show(True)

app.MainLoop() ←メイン・ループを回す
```

▶実行例

```
$ python python_wx.py
```

図4のようにウィンドウを表示するだけのサンプルです．

● GUIを提供する3：PyGTK

GTK＋（The GIMP ToolKit）の機能をPythonから呼び出すライブラリです．GNOMEのルック・アンド・フィールに近いGUIを構築できます．

▶インストール

```
$ sudo apt-get install -y python-
gtk2
```

▶コード例（python_gtk.py）

```
import gtk ←gtkモジュールをインポート
           create_window関数を定義
def create_window():
     window = gtk.Window()
       window.connect('destroy',
                      gtk.main_quit)
     window.show()         ↑終了処理を登録

create_window()
gtk.main() ←メイン・ループ
```

▶実行例

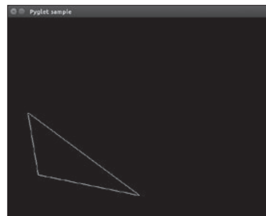
```
$ python python_gtk.py
```

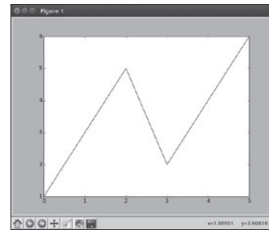
図5のように白い画面が表示されるだけのサンプルです．

● 3DプログラミングVPython

3Dオブジェクトの表示やアニメーションなどを簡単にできます．3行のプログラムで3Dオブジェクトが表示できます．


図6　VPythonの実行例


図7　Pygletの実行例


図8　matplotlibの実行例

▶インストール

```
$ sudo apt-get install -y python-
visual
```

▶コード例（python_visual.py）

```
from visual import *

sphere(color = color.green)
```

▶実行例

```
$ python python_visual.py
```

図6に実行例を示します．特にコードを書かなくても，デフォルトの処理によってマウスのドラッグでオブジェクトを動かすことができます．

● GUIを提供する4：Pyglet

オブジェクト指向のAPIを持つクロス・プラットホームのライブラリです．ゲームやリッチなGUIを持つアプリケーションの開発に使用されます．

▶インストール

```
$ sudo apt-get purge -y python-
pyglet
$ sudo apt-get install python-pip
$ sudo pip install pyglet
```

筆者の環境のUbuntu 15.04のapt-getでインストールされるバージョンは古く問題が発生するため，pipコマンドで最新バージョンをインストールしています．

▶コード例（python_pyglet.py）

```
import pyglet                     ← pygletモジュールのインポート
from pyglet.gl import *

window = pyglet.window.Window     ← ウィンドウの作成
(width=640, height=480, caption=
                "Pyglet sample")
                                  ウィンドウ・イベント処理のためのデコレータ
@window.event
def on_draw():                    ← on_draw関数を定義．デコレータにより，イベント発生時にコールバックされる
    glClear(GL_COLOR_BUFFER_BIT)  ← 画面のクリア
        glPolygonMode(GL_FRONT_AND_
                 BACK, GL_LINE)   ← OpenGLの描画モードの設定
        glBegin(GL_POLYGON)
        glVertex2i(320, 50)
        glVertex2i(75, 100)       ← ポリゴンの描画
        glVertex2i(50, 250)
        glEnd()

pyglet.app.run()                  ← メイン・ループ
```

▶実行例

```
$ python python_pyglet.py
```

OpenGLのAPIを使ってポリゴンを描画するだけの実行例です（図7）．リッチなGUIの作成も手軽にできますが，PythonからOpenGLのAPIを直接的に呼べるのも特徴の一つです．ウィンドウの作成については実質1行で行うことができます．

● 2D/3Dのグラフ描画 matplotlib

2D/3Dのグラフを描画するためのライブラリです．見た目が良いグラフを簡単に描画できます．

▶インストール

```
$ sudo apt-get install -y python-
matplotlib
```

▶コード例（python_matplotlib.py）

```
import matplotlib.pyplot as plt   ← モジュールのインポート
plt.plot([1,3,5,2,4,6])           ← データのセット
plt.show()                        ← グラフの表示
```

▶実行例

```
$ python python_matplotlib.py
```

図8のように2次元の折れ線グラフが表示されます．実質2行のコードでグラフが表示できます．

みつき・ゆうすけ

第3章 定番も最新も! 世界の英知が集結

いま大注目の理由…人工知能ライブラリはPython

佐藤 聖

● ゼロから作ると時間が膨大…人工知能ではライブラリを使う

Pythonで利用可能な人工知能ライブラリを紹介します．一般的に人工知能APIや人工知能クラウド・コンピューティングなどを利用することが多いのではないでしょうか．独自の人工知能アルゴリズムを使用したい場合や人工知能サービスの動作の仕組みを明らかにして使用したい場合もあると思います．通常，ゼロから人工知能システムを開発すると，プログラムのコーディングや実行テストなどを繰り返すため，開発が長期化しがちです．Pythonの人工知能ライブラリを活用すると，開発期間の短縮やコーディングの省力化ができます．

● 人工知能の研究開発にはコーディングしやすいPythonが多用される

人工知能は，統計学を中心にさまざまな分野を組み合わせて実現します．基本的にどのようなコンピュータ言語でも開発できますが，人工知能の研究や仮説検証ではPythonが多用されています．C++などの静的型向け言語と比べて，Pythonは処理速度が遅いというデメリットがありますが，データの型をそれほど意識せずにコーディングできるため，開発者への負荷が少ないというメリットがあります．人工知能を開発する際に，多種多様なデータや数式を利用するため，システム規模が大きくなるほど開発期間やプログラムの品質管理に大きな差が出ます．

● 世界の企業/研究機関の英知＝人工知能ライブラリからはじめるべし

世界中の多くの開発者がPythonライブラリを開発しています．紹介するライブラリの中にもGoogleやマイクロソフトなどの企業，研究機関で開発されたライブラリがあります．高品質のライブラリは，業務やサービス開発にそのまま応用できます．人工知能システムを初めて開発するときなど，何から手をつけてよいか迷いますが，メジャーな人工知能ライブラリにはドキュメントやYouTube動画が多くありますので，最初は情報をたくさん仕入れて，先駆者の知恵を活用するとよいと思います．

● 人工知能ライブラリあれこれ

国内外でポピュラなライブラリを選択しました(**表1**)．

メジャーなPythonの人工知能ライブラリとして，scikit-learn，Theano，TensorFlow，Caffe，Chainerなどがあります．最も利用されていると思われる三つのライブラリを紹介します．

▶機械学習ライブラリscikit-learn

インターネットを検索するとscikit-learnを使用した開発事例が見つかります．機械学習アルゴリズムの学習に利用したり，クラウド型サービスを開発したりすることが容易です．画像認識や自然言語処理などのアルゴリズムが開発しやすく，仮説検証にもぴったりなライブラリです．

▶数値計算ライブラリTheano

Pythonの代表的な数値計算ライブラリです．このライブラリは，実行時コンパイルによる高速化，GPUのサポートによる高速化，自動微分のサポートなどの機能があります．ニューラル・ネットワークやディープ・ラーニングで多用される数式や関数を簡単にコーディングできます．特にGPUのサポートがあると，計算速度をCPUよりも100倍以上速くすることも可能です．

▶Google提供のTensorFlow

Googleが使っている人工知能や機械学習のソフトウェアをオープンソース化したものです．Google製品と同品質の人工知能や機械学習が利用できます．

パソコンからクラウド・コンピューティングまで，さまざまなスケールのシステムが同じコードで動かせるため，企業向けのITシステム開発のプロトタイピングから業務システムの開発まで応用可能です．

● こんなキーワードでもっと見つけられる

ウェブ検索エンジンで，「Python」+「library」に続けて，SVM，Bayes，Machine Learning，Natural Network，

表1　Pythonで利用可能な人工知能関連ライブラリ

番　号	パッケージ名	詳　細	入手先URL
1	BayesPy	ベイズ推定のためのツール	http://bayespy.org/en/latest/
2	breze	Theanoなどによる機械学習のためのツール	https://github.com/breze-no-salt/breze
3	Caffe	ビジョン・アプリケーションにおける機械学習のためのライブラリ	https://github.com/BVLC/caffe
4	Chainer	ディープ・ラーニングのための次世代オープンソース・フレームワーク	https://github.com/pfnet/chainer
5	Crab	推奨エンジン・ライブラリ	https://github.com/muricoca/crab
6	Ffnet	順伝播型ニューラル・ネットワーク・トレーニング・ライブラリ	http://ffnet.sourceforge.net/
7	Gensim	自動的に文書からセマンティック・トピックを抽出するためのライブラリ	http://radimrehurek.com/gensim/
8	Hebel	GPUアクセラレーションによるディープ・ラーニング・ライブラリ	http://hebel.readthedocs.io/en/latest/
9	LibSVM	SVMによる機械学習のライブラリ	https://www.csie.ntu.edu.tw/~cjlin/libsvm/
10	MLPY	機械学習のためのPythonモジュール	http://mlpy.sourceforge.net/
11	mlxtend	データ分析や機械学習ライブラリの拡張ライブラリと補助ライブラリ	https://github.com/rasbt/mlxtend
12	Monte	モンテカルロ・マルコフ連鎖による確率分布サンプリングのライブラリ	https://github.com/baudren/montepython_public
13	Mrec	推薦システムの開発，評価パッケージ	https://github.com/mendeley/mrec
14	neon	高速な成功を実現したディープ・ラーニングのフレームワーク	https://github.com/NervanaSystems/neon
15	Neurolab	シンプルで強力なニューラル・ネットワーク・ライブラリ	https://pypi.python.org/pypi/neurolab
16	Nimfa	非負値行列因子分解のためのPythonライブラリ	http://nimfa.biolab.si/
17	nolearn	ニューラル・ネットワーク・ライブラリ	https://github.com/dnouri/nolearn
18	Peach	ニューラル・ネットワーク，ファジィ論理，遺伝的アルゴリズムなどのライブラリ	https://code.google.com/archive/p/peach/
19	PyBrain	Python用のモジュール式機械学習ライブラリ	http://www.pybrain.org/pages/home
20	pydeep	Pythonのディープ・ラーニングのフレームワーク	http://andersbll.github.io/deeppy-website/
21	Pyevolve	Pythonで書かれた完全な遺伝的アルゴリズムのフレームワーク	http://pyevolve.sourceforge.net/0_6rc1/
22	Pylearn2	Theanoに基づく機械学習ライブラリ	http://deeplearning.net/software/pylearn2/
23	PYMVPA	大規模なデータセットの統計的学習の分析を簡単に行うためのPythonパッケージ	http://www.pymvpa.org/
24	python-recsys	特異値分解による推薦システムのライブラリ	https://github.com/ocelma/python-recsys
25	Ramp	機械学習ソリューションを迅速にプロトタイピングするためのPythonライブラリ	https://github.com/kvh/ramp
26	Reverend	汎用ベイズ識別器	https://sourceforge.net/projects/reverend/
27	scikit-learn	機械学習ライブラリ	http://scikit-learn.org/stable/
28	scipy-cluster	凝集型階層的クラスタリングのためのライブラリ	https://code.google.com/archive/p/scipy-cluster/
29	Spearmint	紙で概説したアルゴリズムに従って，ベイジアン最適化を実行するためのパッケージ	https://github.com/JasperSnoek/spearmint
30	Tensorflow	スケーラブルな機械学習のための「データ・フロー・グラフ」を用いて計算	https://www.tensorflow.org/
31	Theano	Pythonの数値計算ライブラリ	http://deeplearning.net/software/theano/

GA，Deep LearningSVM，Machine Learning，NLPなどの機能で探すともっと見つかると思います．

＊　＊　＊

Pythonはコーディングが容易なため，ライブラリの開発者に多くの学生も含まれます．特にオンライン教育サービスが普及している昨今では，コンピュータ・サイエンス，データ・サイエンス，機械学習，ゲノム，マーケティングなどのコースをパソコンで学習しますので，幅広い分野でライブラリが開発されてます．講座ではさまざまな課題が与えられ，成果を公開することで多くの開発者からフィード・バックを受けられます．成果の公開先としてGoogle CodeやGitHubなどが指定されることがよくあります．

さとう・せい

Appendix 2 はじめての人でも大丈夫!

Python文法超入門

佐々木 弘隆

Python(パイソン)とは軽量なスクリプト言語の一種です．英語ではニシキヘビのことですのでイメージキャラは蛇ですが，元ネタはコメディ番組から来ています．

Pythonのバージョンは2系と3系が出ているのですが，最近の言語には珍しく互換性が低いので2系を使うことが多いようです．ここでも2系のスクリプトに絞って解説します．

構文

● 基本は1行で1処理

スクリプトの1行で一つの処理を記述します．処理は上の行から順番に処理されていきますので，現実世界の演劇などで使われている用語のプログラムと同じように演目(スクリプトでは処理)の並びの通りに進行します．スクリプト・ファイルが終わるか，終了処理(exit構文を書く)でスクリプトの実行は終了します．

● ブロック構造はインデントで表す

スクリプトが単純で小さい場合は処理を並べるだけで済みますが，大きな物を作ると管理が難しく読み難くなってしまいます．多くのプログラム言語では，まとまった処理の集合をブロックという構造でまとめることで読みやすくコンパクトに整理しています．

Pythonの文法上の大きな特徴としては，多くの言語がブロックを表すのに中かっこ{ }やbegin endの句でまとめるのに対して，Pythonはインデントを下げることで表現します．通常は四つのスペース文字が1インデントとなります．

ブロックがないときはインデントは不要です．Hello Python World!と文字表示するプログラムは次の通りです．

```
print "Hello Python World!"
```

拡張子をpyにしたファイルで保存しています(ここではtest.pyとしている)．

```
$ python test.py
```

とすると実行されます．

● セミコロンで区切ると改行せずに書ける

1行が処理単位ですが，セミコロンで区切ると改行せずに書くこともできます．まとめた方が分かりやすいときに便利です．

```
print "Hello "; print "Python World!"
```

● バックスラッシュで複数行に分割

反対に，複数の行に分割するときは行の終わりにバックスラッシュ(\)を書きます．

日本語環境ですと¥(円)記号のキーで打てます(MacはOption＋?)．環境によって，表示が¥や\の場合があります．

```
str = "Hello "¥
      + "Python "¥
      + "World"
```

● コメントの記述には#を使う

#の後ろは全てコメントとなり，処理に影響を与えません．ブロックを表すインデントは元の処理より先頭にスペースを四つ追加します．ブロックが終わったらインデントを戻します．

```
if str == "Python":   #ここのブロックは
      print "It is"   #ここから
      print "Python"  #ここまで
print "Next Process"  #これは別の処理
```

エンコード指定

英数字以外はそのままでは使えません．使う文字のエンコードを指定する必要があります．ファイルの1行目か2行目にエンコード指定を書くルールになっています．

● UTF8でファイルを保存するとき

```
# coding: utf-8
```

● シフトJIS(Windowsなど)のとき

```
# coding: Shift_JIS
```

● EUC-JPのとき

```
# coding: EUC-JP
```

表1　算術演算子の種類
算術演算子は算数・数学とほぼ同じ

記　号	用　途	例
+	加算や正の数の表現に使用	+num や num + NUM2
-	減算や負の数の表現に使用	-num や num - num2
*	乗算に使用. ×(掛ける)の記号の代わり	num * num2
/	除算に使用. ÷(割る)記号の代わり	num / num2
%	剰余(除算の余り)に使用	num % num2
**	べき乗に使用. ** の後が乗数	num ** num2

変数とデータ

● 扱うデータは変数に格納する

＝記号の右辺を左辺に入れるというのが基本的な流れです．変数は英数字と _(アンダバー)の組み合わせですが，最初の文字に数字は使えません．

```
num = 123
_Num2 = 456
```

● 数値表現

10進数以外の数値表現は次の通りです．

```
num2 = 0b0101   # 2進数
num8 = 0o77     # 8進数
num16 = 0xff    # 16進数
```

コンピュータの計算用ビット数を超える桁の数値も扱えます．末尾にLを付けて書きます．

```
longnum = 12345678901234567890L
```

小数は次のように書きます．

```
num = 1.23
num2 = 1.2e3 #指数表記で1.2x10の3乗
```

● 真偽を表す論理値

真偽を表す論理値は次のように書きます．

表2　論理演算子の種類
論理演算子はビットを処理する

記　号	用　途	例
~	ビットの反転. 負数記号のように頭に付ける	~bin
&	論理積. ANDとも呼ばれる. 共に1のときだけ1になる	bin & bin2
|	論理和. ORとも呼ばれる. 共に0のときだけ0になる	bin | bin2
^	排他的論理和. XORとも呼ばれる. 一致してないときだけ1になる	bin ^ bin2
<<	左シフト. ビット全体を指定数だけ左に移す	bin << num
>>	右シフト. ビット全体を指定数だけ右に移す	bin >> num

```
isTrue = True
isOK = False
```

● 文字列

ダブル・クォート(")かシングル・クォート(')を使って文字列を書きます．

```
str1 = "Hello "
str2 = 'Python World!'
```

文字列自体に改行を付けるには\nを使います．

```
str = "Hello¥nPython World!"
```

● データの格納

複数のデータを格納するにはリストを使います．

```
numlist = [1, 2, 3, 4]
strlist = ["Hello","Python","World!"]
```

リストは\記号を使わずに改行して記述できます．

```
strlist = ["Hello",
      "Python",
      "World!"
]
```

変数に変数を代入するには，右辺に変数を置きます．

```
num = 123
num2 = num
```

演算子

データや変数を演算子でつなぐことで演算ができます．

● 算術演算子

表1に示すように，算術演算子は数学とほぼ同じですが，文字の都合上で使う記号が異なります．

● 論理演算子

表2に示すように，論理演算子はビットを処理します．コンピュータや電気は‘0’と‘1’とかON/OFFの組み合わせとか言われていますが，その単位の演算に使います．普段は使う機会が少ないですが，デコード処理やハードウェア処理などで活躍します．

● 代入演算子

代入演算子は，変数の項目で触れた通り右辺を左辺に代入します．右辺は数値や変数が使える他に，演算子を使った処理を行えます．

```
num = 12
num2 = num
num3 = num + 3
```

演算対象と左辺が同じ場合，省略することができます．次の二つは同じ処理になります．

```
num = num + 1
```

```
num += 1
```

● 比較演算子

比較演算子は主に条件判断によって制御の流れを変えるとき（if構文など）に使います．**表3**にPythonスクリプトで使われる比較演算子を示します．演算子の左右の項目を比較し，条件に合っていれば真（True），合わなければ偽（False）になります．

● ブール演算子

ブール演算子は比較結果の論理演算のような使い方ができるため，より複雑な条件を指定できます．**表4**にブール演算子を示します．

● 文字列演算

文字列演算で文字列の操作ができます．

＋演算子を使うと左右の文字列を連結します．"ABC" + "DEF"は"ABCDEF"です．

＊演算子を使うと＊の右で指定した回数繰り返した文字列になります．"ABC" * 2は"ABCABC"です．

［ ］を使うと文字列の特定の文字を取り出せます．str = "ABC"; str[1]だとBが取り出せます．

制御構文/条件分岐

● if構文

if構文を使うと処理を分岐できます．書式は以下の通りですが，elifとelseは必要なければ省略できます．

［書式］

```
if条件：
    条件成立したときの処理
elif:上の条件以外で，新しい条件：
    条件成立したときの処理
else:
    上の条件以外のときの処理
```

［例］

```
if num == 1:
      print "1です"
elif num == 2:
      print "2です"
else:
      print "その他です"
```

● 繰り返し処理

while構文を使うと条件が真の間に繰り返す流れになります．

［書式］

```
while条件:
        条件が成立しているときの処理
```

［例］

```
while num < 10:
      print num
      num += 1
```

表3　比較演算子の種類
主に条件判断によって制御の流れを変えるときに使われる

記　号	意　味
==	等しい
!=	不一致
<	左が小さい
>	左が大きい
<=	左が右以下
>=	左が右以上
<>	不一致
is	等しい
is not	不一致
in	左が右に含まれる
not in	左が右に含まれない

表4　ブール演算子の種類
ブール演算子は比較結果の論理演算的な使い方ができる

記　号	論理判定
and	左右が共に真なら真判定
or	左右が共に偽なら偽判定
not	右の判定を反転

● for構文

for構文を使うと，複数のデータに対して同じ処理を行う流れになります．データ集合にはリストや文字列，ファイルなどが利用可能です．回数指定の繰り返しにはrange()を使うと簡単に実現できます．

［書式］

```
forデータ取得変数inデータ集合
  データに対する処理
```

［例］

```
for data in [1,2,3]:
      print data
for chara in "ABC":
      print chara
for line in open("Textfile.txt")
      print line
for num in range(10):
      print num
```

ループを途中で抜けるにはbreakを使います．

［例］

```
for num in range(10):
      print num
```

コラム1　さすがPythonは超早っ①…世界一短いウェブ・ページ生成コード

佐々木 弘隆

Pythonスクリプトを記載したファイル名をhtmltiny.pyとし，エディタで下記のスクリプトを書き込んで保存します．実行すると世界一短いHTMLコードが生成されます．

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
print ‘ <html></html>'
```

スクリプトの1行目はPythonスクリプトであるという宣言です．2行目はなくても構いませんが，日本語の文字を含むときはエラーになってしまうので記述します．3行目は文字表示をする処理です．

Pythonスクリプトの実行方法は2種類あります．

▶(1) 外部コマンドのPythonインタプリタを実行させて，スクリプト・ファイルを指定します．この方法では1行目の宣言は不要です．

```
$ python htmltiny.py
<html></html>
```

▶(2) chmodコマンドでファイルを実行可能な属性に変更し，Pythonスクリプト名をパス付きで打ち込みます．

```
$ chmod +x htmltiny.py
$ ./htmltiny.py
<html></html>
```

コラム2　さすがPythonは超早っ②…ホントは難しい画像縮小もサクッ

佐々木 弘隆

Python自体でも複雑な計算はできますが，かなり複雑で便利な機能がライブラリの形で提供されています．

ここでは代表的な画像処理ライブラリ「PIL」を用いたPythonスクリプトを**リストA**に示します．画像を扱うには本来難しい処理が必要なのですが，難しい処理はPILが代わりにやってくれます．

次のようにPythonスクリプトを実行すると，画像ファイルを縦横50%に縮小できます．

```
$ chmod +x half.py
$ ./half.py a.jpg
```

ここでの事例では，a.jpgが縦横共に半分に縮小されます．

リストA　画像ファイルを縦横50%に縮小するPythonスクリプト

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from PIL import Image               # PILライブラリを使う準備
import sys                          # OSのシステムを使う準備
img = Image.open(sys.argv[1])       # 元画像の読み込み
print img.filename                  # 画像ファイル名の表示
width = img.size[0]                 # 画像の横幅読み込み
height = img.size[1]                # 画像の縦幅読み込み
nwidth = width / 2                  # 横幅を半分に
nheight = height / 2                # 縦幅も半分に
nsize = nwidth, nheight             # 変換後の画像解像度設定
nimg = img.resize(nsize)            # 画像を縮小
nimg.save(sys.argv[1])              # 縮小画像を書き込み
```

```
    if num == 5:
        print "5で終了"
        break
```

ループの継続にはcontinueを使います．breakは今のデータの処理もループも終わりますが，continueは今のデータの処理だけ終わって，次のデータでループは継続します．

[例]

```
for num in range(10):
    if num == 5:
        print "5は継続"
        continue
    print num
```

関数

似たような処理を何回も書くのは無駄ですので，共通した処理を関数にまとめることができます

[書式]

```
def 関数名(引き数):
    処理
```

[例]

```
def addprint(num, num2):
    print num + num2

def average(num, num2):
    return (num + num2) / 2

addprint(1,2)
addprint(123, 456)
avr = average(1234, 5678)
print avr
```

ささき・ひろたか

ダウンロード・データあります

第4章 画像処理や文字列処理がサクサク！自動更新ウェブ・サーバもサッ

はじめてのPythonプログラミング

佐々木 弘隆

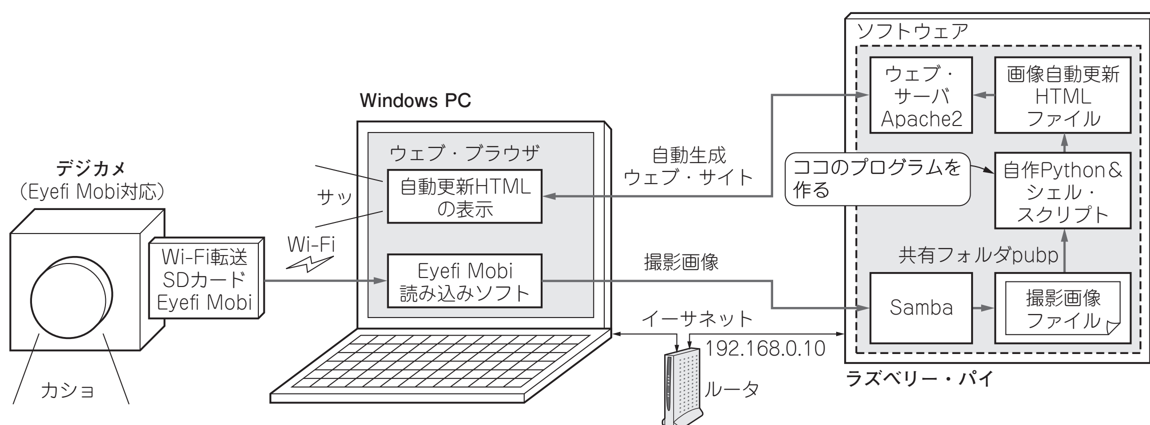


図1 Pythonプログラミング実験の構成
デジカメで撮影した画像をPython＆シェル・スクリプトで成形して，自動的に更新するウェブ・サイトを作ってみる．サーバ・マシンにはラズベリー・パイを使う．写真の加工などは意外と手間がかかるので，面倒なことはコンピュータで自動的にやらせたい

Pythonは，とても使いやすいプログラミング環境です．覚えやすくて書きやすいし，どんどん拡張させることができます．科学技術計算だけでなく，CADや3Dゲーム，ウェブ・サービスなどでも使われています．Pythonプログラムは，Python以外のシステムからでも動かせるので，シェル・スクリプトなどと組み合わせて，電子ブロックのようにプログラムをつなげるだけで，簡単にシステムとして動かせます．

今回は，デジカメで写真を撮影すると画像を自動的に更新するウェブ・サイトを作りながら，Pythonプログラミングを初体験してみます（**図1**）．

実験に必要なもの

ラズベリー・パイは，実売価格が5,000円程度のLinuxが動く小さな定番コンピュータ・ボードです．安価なウェブ・サーバ・マシンとして使えるだけでなく，ライブラリが非常に充実したPythonスクリプトが使えて非常に便利です．今回の実験用ウェブ・サーバ・マシンには，このラズベリー・パイを使うことにします．

そのほか，実験に必要なハードウェアを**表1**に示します．

撮影した画像を取り出すために，デジカメのメモリ・カードをいちいち入れ替えるのは面倒です．なるべく自動で動かしたいので，Wi-Fi無線で画像が送れるSDカードEyefi Mobiを使うことにしました．

大きく以下のような処理の流れになります．

- ▶ステップ1-1：デジカメ撮影画像をSDカード（Eyefi Mobi）に保存
- ▶ステップ1-2：Eyefi Mobiに保存した撮影画像を無線LANでWindows PCに転送
- ▶ステップ2：Windows PCに転送されてきた撮影画像をラズベリー・パイの共有ディレクトリに転送
- ▶ステップ3：ラズベリー・パイに転送されてきた撮影画像から自作プログラムでウェブ・ページを作成
- ▶ステップ4：自動作成したウェブ・ページをWindows PCのブラウザで表示して確認

表1　実験に使うハードウェア

項　目	用　途
ラズベリー・パイ	ウェブ・サーバ・マシン
microSDカード(8Gバイト程度)	ラズベリー・パイ起動用
Eyefi Mobi	撮影画像用のWi-Fi転送機能付きSDカード(http://jp.eyefi.com/products/mobi)．撮影した画像を取り出すために，デジカメのメモリ・カードをいちいち入れ替えるのは面倒なので今回使用
デジカメ	撮影用．画像転送にEyefi Mobiを使いたい場合は対応機種でないといけない(http://jp.eyefi.com/camera)
Windows PC	Eyefi Mobiを使う場合は，読み込みソフトウェアが動かせないといけない．ブラウザでラズベリー・パイで自動生成したウェブ・ページを確認するためにも使う

準備

● 準備1：デジカメ撮影画像をWi-Fi付きSDカードEyefi Mobiを使ってPCに転送できるようにする

まずは写真撮影側の準備をします．

デジカメを用意します．

Eyefi Mobi対応機種のデジカメ(結構多い)を用意して，Windows PCにEyefi Mobiの読み込みソフトウェアをインストールします．

撮影してみて，PCに写真が転送できていれば準備完了です．

● 準備2：撮影画像保存用にラズベリー・パイに共有ディレクトリを作る

Windows PCからRaspbian(Linux)へファイルを送るにはSambaを使います．今回は，共有ディレクトリ名はpubpiとしています．

ラズベリー・パイにログインしたらディレクトリを作成します．

```
$ mkdir pubpi
$ chmod +w pubpi
```

次にSambaの準備をします．



図2　ラズベリー・パイ上の共有フォルダ(今回はpubpi)にネットワーク・ドライブの割り当てをしてデジカメ撮影画像を直接保存できるようにする

```
$ sudo apt-get install samba
```

インストールが完了したら共有ディレクトリの設定をします．以下のコマンドを実行して，簡易テキスト・エディタnanoなどを使い，Samba設定ファイルを編集します．

```
$ sudo nano /etc/samba/smb.conf
```

ファイルの最後には以下を追加します．

```
[pubpi]
comment=publicpi
path=home/pi/pubpi
public=Yes
read only=No
writable=Yes
guest ok=Yes
force user=pi
directory mode=0777
create mode=0666
```

● 準備3：ウェブ・サーバ・マシンとして使うラズベリー・パイのIPアドレスを設定する

ラズベリー・パイは，標準ではDHCPによってIPアドレスが自動割り振りされるので，動作を確認するには不便です．ここでは固定IPアドレスの設定をします．

LAN内のIPアドレスが192.168.0.nの範囲で設定される環境で，ラズベリー・パイには192.168.0.10を割り当てます．DHCPの設定ファイルをテキスト・エディタで開き，

```
$ sudo nano /etc/dhcpcd.conf
```

以下を追加します．

```
interface eth0
static ip_address=192.168.0.10/24
static routers=192.168.0.1
static domain_name_servers=192.168.0.1
```

再起動して設定を有効にします．

```
$ sudo reboot
```

● 準備4：Windows PCで共有フォルダをネットワーク・ドライブ割り当てする

再起動を完了しましたら，Windows PCのエクスプローラーを開いて，アドレス¥¥192.168.0.10を入力します．

pubpiが表示されたら，右クリックからネットワーク・ドライブの割り当てができたら準備完了です(図2)．

● 準備5：Eyefi Mobiの写真取り込み先を共有フォルダに変更する

ファイル共有ができましたら，Eyefiの写真取り込

みディレクトリを共有ディレクトリに変更します．

PCにインストールした「Eyefiオプション」を起動して，高度な設定を開きます．

［転送先の設定］→［フォルダの場所］→［写真］の設定を用意した共有フォルダに変更します．

また，サブフォルダを作成のチェック・ボックスは外しておきます．

デジカメ撮影画像がラズベリー・パイの共有フォルダに保存できるようになったか，写真撮影して確認しておきます．

```
$ ls pubpi/
_Dxxxx.JPG
```

撮影された写真ファイルが，共有フォルダ（今回はpubpi）で確認できれば，ここまでの準備は完了です．

● 準備6：ウェブ・サーバ・ソフトApacheをラズパイにインストールしておく

ウェブ・ページの作成の前に，ウェブ・サービスを準備します．いろいろなサービスがありますが，定番で情報が探しやすいApacheを使います．

```
$ sudo apt-get install apache2
```

Apacheのインストールが完了したら，PCのブラウザでアクセスしてみます．アドレス・バーに，192.168.0.10を打ち込み，**図3**のような画面が出たら準備完了です．

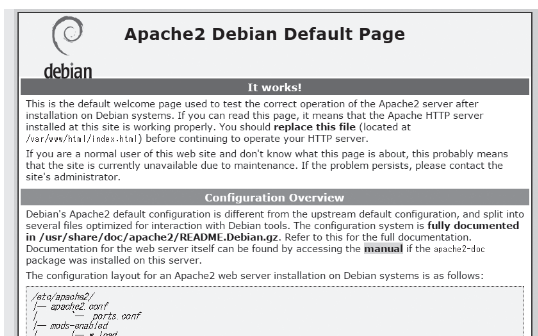


図3　ウェブ・サーバ・ソフトApache2をラズベリー・パイで動くようにしておく

ウェブ・ページ自動生成プログラムの作成

ここからはいよいよ，Pythonスクリプトを使ってウェブ・ページの自動生成ソフトウェアを作成していきます．

● 基本方針…細かい機能プログラムを電子ブロック的に組み合わせてやりたいことを実現する

Pythonスクリプトは非常に手短にプログラムを記述できます．例えば，今回作成した画像のサイズ変更スクリプトも20行程度で済みます．

また，ファイル名の取得などはLinuxの`ls`コマンドの方が便利そうですし，出力はパイプで受け渡した方が楽な場合もあります．その場合はPythonではなく，シェル・スクリプトを選ぶ方がよさそうです．

そこで今回は，細かな機能プログラムをPythonスクリプトやシェル・スクリプトで作成し，それらを電子ブロック的に組み上げることで，プログラムを作成してみようと思います．

Pythonやシェル・スクリプトを使うと，いろいろな機能を短いプログラムで実現できることを体感できるのではないかと思います．

● 作成するプログラム・ブロックを整理する

プログラムを組み上げる前に，何をするのか整理して，使う部品の選択をしなくてはいけません．

撮影画像を発見してウェブ・ページを自動作成するのに必要なソフトウェア・ブロックには，次のようなものがあります．

- 写真一覧取得
- 写真のコピー
- 写真の縮小
- HTML骨組み作成
- 写真のHTML作成
- 写真差分判断

これに，ウェブ・ページ（要はHTMLファイル）を作成するプログラムが必要です．これらのソフトウェア・ブロックを用意していきます．

● その①…写真の縮小ブロック

デジカメのオリジナル撮影写真はサイズが大きすぎるので，ブラウザで表示できるように縮小します．

また，ブラウザに表示するときには，画像が同じサイズだと便利です．カメラの向きを縦や横に変えたときでもそのサイズに収まる範囲で縮小すればつぶれた写真にならなくて済みます．

機能1：縮小したときの最大の大きさを指定できるようにする．

　スクリプト実行時にパラメータを受け取れるようにします．

機能2：縦横比を変えずに指定サイズに収まる縮小率を計算する．

　縦横比の違いで処理を二つに分けて，縮小率を求めます．

プログラムを**リスト1**（`resize.py`）に示します．

● その②…写真一覧取得ブロック

画像ファイルの一覧取得には，内部コマンドの`ls`が使えそうです．内部コマンドだってブロックに変わ

リスト1　写真縮小ブロック(resize.py)

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from PIL import Image
import sys
img = Image.open(sys.argv[1]) # 元画像の読み込み
reducew = int(sys.argv[2])     # 縮小横幅限界値の読み込み
reduceh = int(sys.argv[3])     # 縮小縦幅限界値の読み込み
width = img.size[0]            # 画像の横幅読み込み
height = img.size[1]           # 画像の縦幅読み込み
if (width / height) >= (reducew / reduceh):
                               # 画像の方が指定より横長の時
        nwidth = reducew       # 横幅は指定値に
        nheight = height * reducew / width
                               # 縦幅は横幅の縮小率に補正
else:                          # 画像の方が指定より縦長の時
        nheight = reduceh      # 縦幅は指定値に
        nwidth = width * reduceh / height
                               # 横幅は縦幅の縮小率に補正
nsize = nwidth, nheight        # 変換後の画像解像度設定
nimg = img.resize(nsize)       # 画像を縮小
nimg.save(sys.argv[1])         # 縮小画像を書き込み
```

りはありません．ですが，ディレクトリ名(この場合pubpi/)が邪魔なので，ファイル名だけを取り出すブロックが必要そうです．

```
$ ls pubpi/
pubpi/_D001.JPG
```

これはシェル・スクリプトの方が簡単そうなので，こちらで作ります．プログラムを**リスト2**に示します．

このブロックを先ほどのlsとパイプで接続すると，

```
$ls pubpi/ | delpath.sh
_D001.JPG
```

となります．一つのブロックでダメなら二つのブロックを組み合わせれば何とかなります．

リスト2　シェル・スクリプトでフォルダ内のファイル名だけを取り出すブロック(delpath.sh)

```
#!/bin/sh
if [ -p /dev/stdin ]; then
        cat - |while read line
        do
                fname="${line##*/}"
                echo $fname
        done
else
        exit 1
fi
```

● その③…写真のコピー・ブロック

これは内部コマンドのcpがそのまま流用できます．

● その④…HTML骨組み作成ブロック

HTMLの構成を考えると，メインの写真の前にヘッダが必要で，最後にも終端処理が必要になります．

そこで，二つのブロックを作成します．

前を**リスト3(a)**(htmlpre.py)に，後ろを**リスト3(b)**(htmlpost.py)に示します．

htmlpre.pyは，<html>から<body>までを出力し，パラメータからタイトルを指定できます．

htmlpost.pyは，</body>から</html>まで出力します．

この二つのブロックを使えばメイン・コンテンツなしのHTMLファイルのひな形を作成できます．

● その⑤…写真のHTML作成ブロック

画像指定のタグにして，パラメータで受け取ったファイル名を書き込みます．プログラムを**リスト3(c)**(htmlbody.py)に示します．

● その⑥…写真差分判断ブロック

毎回すべての写真を加工するのは無駄なので，新しい写真ファイルだけを取り出します．

元写真一覧や加工済み写真一覧は，写真一覧取得ブロックで作れるので，二つのファイルを比較して片方に追加されたものだけを取り出します．

プログラムを**リスト4**に示します．

```
$ ./dirdiff 全ファイル・リスト 削除ファイル・リスト
```

最初の指定ファイルから，二つ目の指定ファイルと同じものを除いて出力します．

● その⑦…つなぎ込みブロック

主だったソフトウェア・ブロックはそろいましたので，ブロックを接続します．接続にはシェル・スクリプトを使います(**リスト5**)[編集部注]．つなぐ方法はパイ

リスト3　HTMLファイルを自動作成するPythonプログラム・ブロック

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import sys
print '<html> <head> <title>', sys.argv[1],' </
                             title> </head> <body>'
```

(a) 前部分(htmlpre.py)

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import sys
print ' </body> </html>'0
```

(b) 後部分(htmlpost.py)

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import sys
print sys.argv[1]+'<p> <img src="'+sys.argv[1]+'"
                       alt="'+sys.argv[1]+'"> </p>'
```

(c) 本体部分(htmlbody.py)

編集部注：シェルについては第2特集で紹介します．

リスト4　更新された写真だけを取り出すPythonプログラム・ブロック(dirdiff.py)

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import sys
sdat = open(sys.argv[1]).readlines()
                               # オリジナル・リスト読み込み
ddat = open(sys.argv[2]).readlines()
                               # 除外リスト読み込み
for line in sdat:              # オリジナルを1データずつ取り出し
    match = 0                  # 除外判定
    for delkey in ddat:        # 除外データを1データずつ取り出し
        if line == delkey:     # 除外対象判定
            match = 1          # 除外確定
            break              # 次のオリジナル判定に
    if match == 0:             # 除外対象外判定
        print line,            # 結果出力
```

リスト5　つなぎ込みブロック(dirupdate.sh)

```
#!/bin/sh
ls pubpi/*.JPG | ./delpath.sh > piclist
                         # Windows共有ディレクトリの画像リスト取得
ls picwork/*.JPG | ./delpath.sh > worklist
                         # 作業ディレクトリの画像リスト取得
./dirdiff.py piclist worklist > newlist
                         # 共有に追加された画像リスト取得
./dirdiff.py worklist piclist > dellist
                         # 共有から削除された画像リスト取得
echo 'Append process'
filename=newlist         # 追加処理
cat ${filename}|while read line
do
   echo ${line}:Copy
   cp pubpi/${line} picwork
                         # 共有から作業ディレクトリにコピー
   ./resize.py picwork/${line} 320 240
                         # サイズ変更
   cp picwork/${line} /var/www/html/
                         # ウェブ領域にコピー
done

echo 'Delete process'
filename=dellist         # 削除処理
cat ${filename}|while read line
do
   echo ${line}:Delete
   rm picwork/${line}    # 削除
done
echo 'HTML process'
ls picwork/*.JPG | ./delpath.sh > htmllist
                         # 確定作業ディレクトリの画像リスト取得
./hpre.py インターフェース来訪者様 > index.html
                         # HTMLヘッダとタイトル出力
filename=htmllist        # 削除対象リスト作成
cat ${filename}|while read line
do
   echo ${line}:HTML
   ./hbody.py ${line} >>index.html
                         # 写真コンテンツ追加
done
./hpost.py >> index.html
                         # HTMLフッタ出力
cp index.html /var/www/html/
                         # HTMLをウェブ領域にコピー
```

プやリダイレクト，パラメータなどいろいろあります．

ブロックにもシェル・スクリプトを使いましたが，つなぐときもシェル・スクリプトを使いました．つまり，ブロックをつなぎ合わせたシェル・スクリプトも新たなブロックになります．このブロックはさらに別の起動用ブロックからつながります．

リスト6　起動ブロック(bgtask.sh)

```
#!/bin/sh
while :
do
        ./dirupdate.sh    # ブロック起動
        sleep 1m          # 1分間スリープ
done
```



図4　動作確認…PCのブラウザにラズベリー・パイのIPアドレス(今回は192.168.0.10)を入力すると撮影したての写真が載っているウェブ・ページが出てくる

● その⑧…起動ブロック

最後に起動ブロックを作ります．このブロックがなくてもHTMLの作成はできますが，自動的に動くウェブ・サーバに仕立てます．一定時間ごとに動かすようにします(**リスト6**)．

ラズベリー・パイをこの処理専用にするならともかく，コマンドが打てないのは不便なことがあるので影でこっそり動いてもらいます．

```
$ bgtask.sh &
```

と，後ろに「&」記号を付けて実行するとコマンドの受付待ちに戻ります．もちろんコマンドを打てますが，キチンと動いていますので安心です．「&」記号を付けなくても(`$ bgtask.sh`)動きますが，いったん動きますと強制終了させるまで延々と動いてしまいます．

● 動作確認

デジカメで撮影すると，Eyefi Mobiが写真を転送します．ラズベリー・パイの共有ディレクトリにしっかり送られるはずです．起動ブロックが1分ごとに起



図5　タイトル画面を追加したウェブ・ページのひな形をDreamweaverで作成しておいた

リスト7　自動作成するウェブ・ページの見栄えをよくするように改造

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import sys
print '<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML
4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html> <head> <meta http-equiv="Content-Type"
content="text/html; charset=utf-8"> <title>', sys.
argv[1], '</title> <style type="text/css"> body {
background-color: #005506; } </style> </head>
<body> <img src="interfaceland.png" width="773"
height="194" alt=""/><p>'
```

（a）前部分（htmlpre(2).py）

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import sys
print '</p> </body> </html>'
```

（b）後部分（htmlpost(2).py）

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import sys
print '<img src="'+sys.argv[1]+'" alt="'+sys.
argv[1]+'">'
```

（c）本体部分（htmlbody(2).py）

動するので，新しい画像ファイルを転送して変換しています．

PCのブラウザで確認すると撮影したての写真が載っているウェブ・ページが出てきました（図4）．

見栄えや使い勝手のよいサイトに仕立てる

●その①…見栄えのよいサイトに仕立てる

最低限動くようになったので，最初の目的は果たしました．しかし，見た目には不満があります．ブラウザ上の見た目は写真が縦に並んでいるだけで，何のページか分かりません．デジカメのファイル名が見えてもうれしいわけでもありません．全部縦にする必要もありません．改良の余地があります．

ブラウザが受け取っているのはHTMLファイルなので，見栄えの良いHTMLを作成します．HTMLなら簡単に作成できるエディタが豊富にあるので，それでサクッと作ります．

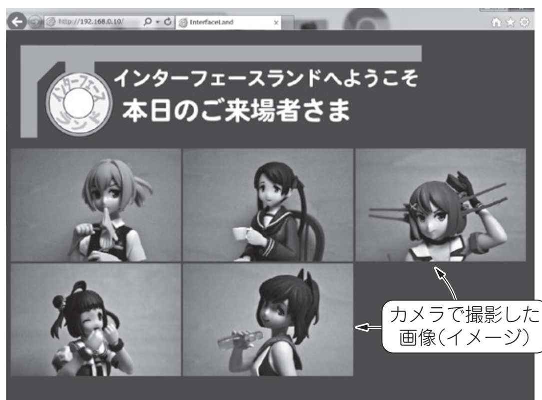


図6　インターフェース編集部への来場者をウェブで自動的に紹介…のイメージで見栄えのよいサイトに仕上げた



図7　自動更新した画像を消したくなる事情があるかもしれない

筆者がDreamweaverで作った画面を図5に示します．画像は/var/www/html/にコピーします．

このHTMLをソフトウェア・ブロックに取り込みます．

htmlpre.pyとhtmlpost.pyのHTML文を置き換えます（リスト7）．

写真の縦並びとファイル名表示を書き換えます．

完成したウェブ・ページを図6に示します．

● その②…写真を簡単に消せるようにしておく

作り手が良かれと思っていても，外部要因で思わぬ対処に迫られることは現実でもよくあります．「握手はいいけど，写真は事務所を通してね」というお客さまのクレームが入りました（図7）．

そうなったときは，消して欲しい写真を選んで消したら自動的にウェブ・ページも写真が消された状態にします．新しい写真を選ぶブロックのつなぎ方を反対にすると消された写真が分かるので，その写真をウェブ用のファイルから消します．

このように，Pythonスクリプトとシェル・スクリプトを適材適所に使って組み合わせれば，いろいろな機能を簡単に実現することができます．機能やデザインを自分好みにどんどん変えることができます．

ささき・ひろたか

Appendix 3 マイコンからクラウド，スパコンまで!? 科学技術計算はコレで決まり!

この先10年は定番!? Pythonが広まっている理由

高橋 知宏

Pythonはすでに20年以上の歴史があるスクリプト言語ですが，最近特に利用者が増えてきているようです．似たようなスクリプト言語は，他にもあるにもかかわらず，なぜPythonの利用が増えてきているのかを私見を交えて考察してみたいと思います．

▶理由1：これがデカい! 数値計算標準ライブラリNumPy/SciPyの存在

最も重要なポイントは数値計算系のライブラリが充実していることにより，科学，工学系，そしてファイナンスの分野で多く使われるようになったことにあると思います．NumPyという基本的な数値計算系のライブラリの存在により，Pythonにおける基本的な数値計算の様式が定められました．そこからNumPyの存在を基礎として，統計や数多くの科学技術計算を網羅したSciPyというライブラリが整備されています．そしてMatplotlibという高品質のグラフ描画ライブラリが提供されています．さらに時系列データを扱うためのライブラリPandasがあります．基本的な処理については，これを使えばOKというライブラリが確定している，このことが重要だったのではないかと思っています．

▶理由2：科学技術系のライブラリが充実

もう一つは，応用ライブラリがとても充実していることがあります．科学技術系の分野であれば，キーワードを元に探せばなんらかの実装を見つけることができます．とても簡単に使えるパッケージ管理があるため，pipというコマンドを1行入力するだけで，ライブラリの検索，そしてインストールを簡単に行うことができ，すぐに利用できます．

pypi.python.orgというサイトによると，驚くことに85,000以上のライブラリが提供されています[ちなみにRuby向けのgemでは7,000個強(決して少なくはない)のライブラリが登録されていますが1桁の差があります]．

なぜ応用(アプリケーション)ライブラリが数多く存在しているのでしょうか．どうも計算機やソフトウェアを専門としていない，他の応用分野に従事している研究者やエンジニアが日常的に使っている言語がPythonであるようです．

米国の大学などでは，コンピュータの専門家向けではないプログラミング・コースではPythonが使われることが多くなってきているとのことです．このことから幅広い分野において，研究者やエンジニアが使うプログラミング言語の第1選択としてPythonが選ばれているようです．彼らが仕事で必要なプログラムを作り，それを汎用的に使えるようライブラリにまとめ，公開，共有されています．そのようなコミュニティが分野ごとに数多く存在しますが，それらを横断的に1カ所にまとめるウェブ・サイトとしてpypi.python.orgが機能しています．目的のプログラムをpipコマンドなどで探すと，たいてい何かを見つけられます．

▶理由3：高性能な専用アルゴリズムも使える

ある程度の規模の仕事をしようとすれば，行列計算，すなわち線形代数が重要です．この分野では高性能なアルゴリズムが実装されたBLASやLAPACKといったライブラリが存在します．これらはFortranで実装されているのですが，NumPy/SciPyを使うことで，Fortranで実装されていることを全く意識せずにPythonから利用することができ，長年にわたって鍛え上げられたその能力を享受できるのです．また近年はGPUを使用して高性能な処理が行われることも多いですが，NumPyで記述したコードをGPUを使って実行できるような環境(NumbraPro)もあります．また最近特に重要な分野として，ディープ・ラーニングや機械学習が盛んに研究，応用されていますが，scikit-learnライブラリで多くの機械学習アルゴリズムが提供され，すぐに利用可能です．また，CaffeやTheano，Chainer，TensorFlowといった話題のディープ・ラーニング向けフレームワークの多くはPythonからの利用が前提だったり，容易だったりします．

▶理由4：マイコンからスパコンまでどのコンピュータでも使える

ちょこっとしたテキスト処理から，GUIのあるデスクトップ・アプリケーション，クラウド・サーバのサービスなどもごく普通にPythonで実装されています．利用可能な環境は幅広く，スパコンなどのハイエンドからクラウド上のサーバ実装，PCはもちろん，小さいところではラズベリー・パイ，最近ではMicroPythonといったマイコン用の実装まで提供されています．

Pythonは，少なくともこの先10年くらいは，広く使われるコンピュータ言語として優位な地位を占めるだろうと考えています．

たかはし・ともひろ

Web公開データあり

第5章 対話型グラフ表示で確認しながら楽々チューニング!

Python科学技術ライブラリを使った無線機のディジタル・フィルタ設計

高橋 知宏

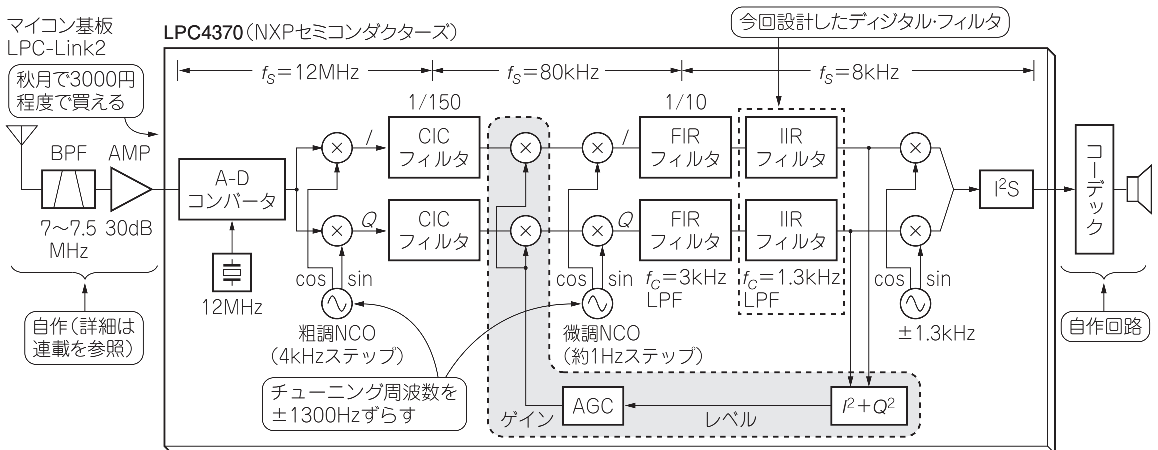


図1[(2)] **今回の設計ターゲット…ソフトウェア信号処理無線機のIIR型ローパス・フィルタ**
本誌で11回(2015年10月号〜2016年8月号)にわたって紹介してきた連載記事「高速ワンチップ・マイコンではじめるソフトウェア無線」の第8回で紹介したソフトウェア信号処理7MHz帯アマチュア無線SSB受信機の信号処理フロー図

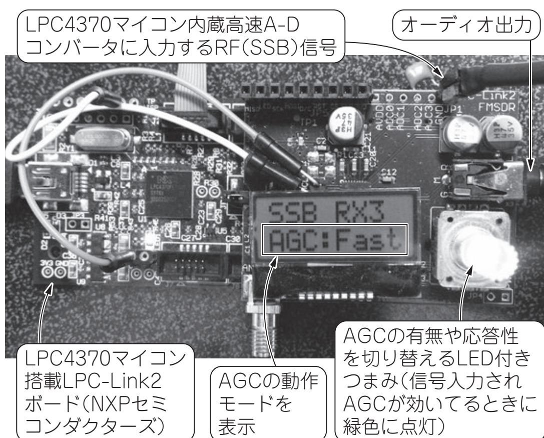


写真1[(2)] **筆者が作成したソフトウェア無線機**

● Pythonの科学技術ライブラリを使って無線機のディジタル・フィルタ設計に挑戦

信号処理においてフィルタは基本的な処理です．ディジタル・フィルタの設計は手計算では大変難しく，ツールを使って設計を行うことが一般的です．ところがディジタル・フィルタの設計に利用可能なツールはいろいろとあり，選択に迷ってしまうほどです．

筆者が作成したソフトウェア信号処理の7MHz帯アマチュア無線SSB受信機では，ローパス・フィルタ(低域通過フィルタ：LPF)として，IIR方式のディジタル・フィルタを使用しました[(2)](**図1**，**写真1**)．

このディジタル・フィルタの設計にはPythonを使いました．Pythonはプログラミング言語であって，フィルタ設計用のツールというわけではないのですが，フィルタ設計に使えるライブラリ(scipy.signal)がほぼ標準的な形で提供されています．ただし，グラフを描画したりするのはそれほど簡単ではありませんので，使いこなすためには，ある程度はコードを読み書きできることが必要です．正直なところ，コードを書くことに慣れていないと少し難しいところもあります．ここでは具体的に詳しく手順を紹介したいと思います．

また，フィルタ設計の理論には触れませんが，成書が多く出版されていますので参照できます．

さらに，SSB受信機に使用したIIRフィルタの設計を取り上げます．FIRフィルタの設計法については参考文献(1)でも紹介しています．

準備

● 準備1：Pythonの実行環境のインストール

初めにPythonの実行環境を用意します．実行環境だけではなく，いろいろなライブラリを合わせて用意する必要があります．このために専用のパッケージ・ソフトウェアが用意されています．今回は簡単にAnacondaというツールを使用する方法を紹介します．

Anacondaは，Windows，Mac OS X，Linuxの各プラットフォームの32ビットと64ビットどちらにも対応しています．Pythonのバージョンとして，2系(2.7)と3系(3.5)を選ぶことができますが，今回は3.5を使います．インストーラもコマンドライン版とグラフィカル版が好みに合わせて選択できます．

インストーラはウェブ・サイト(`https://www.continuum.io/downloads`)からダウンロードできます．使っているプラットフォームと好みに合わせてダウンロードして実行します．

Anacondaでインストールした場合，必要なパッケージはほとんど標準で用意された状態になっています．参考に**表1**に必要なパッケージを示します．

● 準備2：対話型環境Jupyterを起動する

Pythonでの設計作業を対話的に行うために，Jupyter Notebookという環境を使用します．Jupyter Notebookは，ウェブ・ブラウザを使って，Pythonの式を入力して即座に実行したり，結果をグラフに表示したりといった作業を対話的に行うことができてフィルタ設計に非常に便利です．またメモや解説のテキストを一緒に保存できます．

ターミナルやコマンドラインから，

```
jupyter notebook
```

と入力し実行します．するとブラウザの画面が開いてJupyter Notebookが利用可能になります．コマンド・ラインはサーバとして動作しますので，そのままおいておきます．

ブラウザから`http://localhost:8888/`にアクセスして開いてもOKです．

画面には，**図2**のファイル一覧が表示されているはずです．ここでNewボタンを押しPythonを選択します．そうすると別の画面が開いて，空のセルが一つ表示されている状態になります．セルをクリックして，Pythonの任意の式を入力して実行することができます．

例えば，“1+2”を入力し，[Shift]＋[Enter]を押します．そうすると3が表示されます．

[Enter]だけではセル内で改行されるだけなので，式を実行するには，[Shift]＋[Enter]または[Ctrl]＋[Enter]とします．このセル内で関数やクラスを定義することも可能です．ライブラリを使用するには`import`文を使います．

● 準備3：グラフを描いてみる

Jupyter Notebookでは，同じ画面内でグラフを作成することも可能ですが，少しおまじないが必要です．セル内に**図3**の内容を入力して，[Shift]＋[Enter]しておきます．

`%matplotlib inline`は，matplotlibで描画するグラフを画面内インラインで表示することを指示しています．

`import`文はライブラリを取り込んで，短い名前と空間名を与えています．例えば，`import numpy as np`とは，numpyというパッケージを取り込んでnpという名前でアクセスできるようにしています．NumPyの定義するsin関数は，`np.`という接頭辞を付けて，`np.sin`という名前で利用することができ

表1　無線機のディジタル・フィルタ設計に必要なPythonライブラリはAnacondaパッケージを使用すれば最初から用意されている

名　称	内　容
NumPy	数値計算を効率化するライブラリ
SciPy	科学計算ライブラリ
matplotlib	グラフィック・プロット・ライブラリ
Jupyter	プログラムを実行したり，実行結果を記録したり，ノートのように使えるウェブ・ベース対話型環境

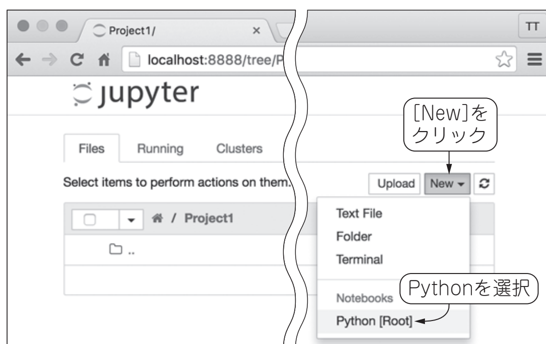


図2　ディジタル・フィルタ設計環境として対話型 Jupyter Notebookを使う
起動するとファイル一覧画面が開く．[New]を押してPythonを選択すると，新規のノートが作成できる

```
In [1]: # グラフをインライン表示するための指定
        %matplotlib inline
        # 必要なライブラリをインポート
        import pylab as pl
        import numpy as np
        import scipy.signal as signal
```

図3　対話環境Jupyter Notebook上でグラフを描画できるようにおまじないを入力しておく

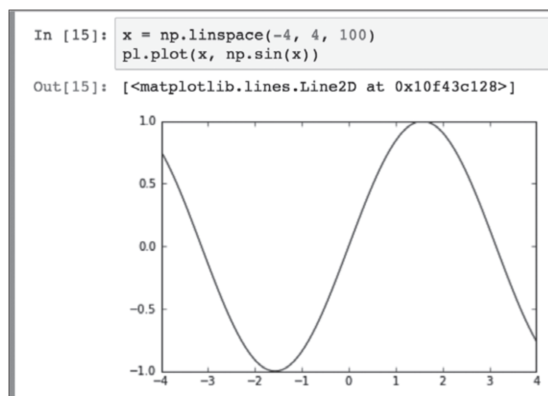


図4　グラフ描画ライブラリmatplotlibで三角関数のグラフを描画してみる

るようになります．

図4のように入力してみます．これの意味は，xが－4～4の範囲で三角関数sin(x)を描画しています．NumPyを使っている部分が不思議に見えるかもしれません．xがNumPyの配列になっていて，それをnp.sin(x)として評価すると，xの要素すべてについてsinを計算し，その結果がまた配列になります．その配列をグラフに描画しているのです．

● 慣れたらなんて書きやすい…Python計算記述

配列をいきなりsinの引き数にしてしまうところに違和感を抱くかもしれませんが，NumPyは同じ計算をまとめて適用する操作（いわゆるベクトル演算）がシンプルに書けるようにデザインされています．配列の要素を操作するためにfor文で回したりmapを適用したりする必要がありません[注1]．

無線機用SSB帯域制限ディジタル・フィルタを設計する

● ステップ１：フィルタの仕様を決める

設計するフィルタはローパス・フィルタです．フィルタを設計するためには諸条件からパラメータを決めていく必要があります．表2に今回の設計パラメータを示します．

可能なサンプリング・レートは信号処理の過程の設計ですでに決まります．一方，リプルや阻止量は，フィルタ次数やフィルタ特性（急しゅんさ）とのトレードオフになります．通過特性のうねりであるリプルは小さい方が，また阻止すべき周波数帯の漏れを減らすために阻止量は大きい方がよいのですが，そのためにはフィルタの次数を増やす，またはフィルタの肩の特性が甘くなる，などの兼ね合いがあります．

設計の結果を確認して，必要であれば設計パラメータを修正した上で再設計してみることになります．フィルタの設計ではこのプロセスを繰り返します．

注1：これに慣れてしまうと，NumPyのない他の言語環境を選択することが減って，その結果としてPython中毒化するわけです．

表2　7MHz帯アマチュア無線SSB受信機用に設計するディジタル・フィルタの仕様

項　目	内　容
サンプリング周波数（f_S）	8000Hz
カットオフ周波数（f_C）	1300Hz
フィルタの形式	楕円フィルタ
リプル	1dB
最小減衰量	40dB

● ステップ２：フィルタ設計関数で係数を求める

フィルタの設計は，フィルタ・パラメータをフィルタ設計関数に与えて，伝達関数を求め，その特性を確認するという流れになります．

Pythonの標準ライブラリSciPyには，信号処理関連の関数群がまとめられているscipy.signalというパッケージが用意されています．この中にフィルタを設計する関数がありますのでこれを使います．いくつも種類があるので迷ってしまいますが，試してみて目的が達成できればOKと思います．

MATLABと似た関数も用意されているようですので，MATLABに慣れた人ならそれを使ってもよいと思います．

今回設計してみるフィルタは，急しゅんな特性を実現可能な楕円フィルタです．scipy.signal.ellip関数を使って設計を行います．さっそくJupyter Notebookで試してみましょう．先ほどと同様，Jupyter Notebookを開き，最初のセルにおまじないを入力して実行しておきます（図3）．

フィルタの次数を6，リプルを1dB，最小減衰量を40dBとします．肝心なカットオフ周波数は，ナイキスト周波数との比で与えます．サンプリング周波数が8000Hzですから，ナイキスト周波数はその半分の4000Hzです．カットオフ周波数がf_C＝1300Hzなので，1300.0/4000として与えています．整数ではなく実数として与えるため1300.0という形式にしてあることに注意してください（整数で除算すると0になってしまう）．最後にローパス・フィルタであることを示すために'low'という文字列を指定します（図5）．

● ステップ３：特性をグラフに描いて確認する

設計の結果はb，aという二つの配列として得られます．これは図6のような伝達関数の分子と分母の多項式の係数です．このままではどんなフィルタを生成

```
In [2]: b, a = signal.ellip(6, 1, 40, 1300.0/4000, 'low')
        b, a

Out[2]: (array([ 0.03017533, -0.01659437,  0.06473288, -0.01819699,  0.06473288,
                -0.01659437,  0.03017533]),
         array([ 1.        , -3.39619492,  6.10406274, -6.64460001,  4.64203379,
                -1.94403401,  0.39405421]))
```

図5　フィルタ設計関数を使ってローパス・フィルタを設計する
結果は伝達関数の多項式係数として得られる

したのか分かりませんので，これをグラフ化します．

伝達関数から周波数特性を得るためにscipy.signal.freqz関数が用意されています．これは引き数として係数を受け取り，周波数特性を返します．wとhという組になっていますが，wが周波数（サンプリング周波数が2πに相当するスケール），hが振幅（リニア）です．振幅は複素数ですので位相の情報も持っています．

これをグラフ化するにはpl.plot関数に，横軸をHz単位にスケールした周波数，縦軸をdB単位に変換するためhの絶対値（abs）の常用対数（log10）をとり20を乗じたものを与えます．そうすると**図7**のようなグラフが得られます．

もし望みの特性となっていない場合は，次数やリプル，阻止量，カットオフ周波数などのパラメータを調整して，グラフを確認するという一連のプロセスを，望む特性が得られるまで繰り返します．セルに与えたパラメータを変更して，再実行するだけでOKです．セルをコピー＆ペーストして別のセルで実行すれば，特性を見比べることも容易です．

● ステップ4：精度が出しやすいように2次カスケード・フィルタの形に分解した係数を得る

特性が確認できたら，これを実装するための用意をします．ところで，高い次数のフィルタをそのまま実装するといろいろと不利な点があります．例えば，係数の誤差にクリティカルになったり，計算精度が足りず飽和したりするなどの現象が発生してしまいます．そこで，フィルタを構成する形式として，次数の低い2次フィルタを順に並べた（カスケード）形とすることで，不都合を生じにくくするテクニックが使われます．

今回の例では，伝達関数の分子と分母は6次の多項式として得られています．これを2次フィルタに分解するということは，6次多項式を因数分解して，三つの2次多項式の積とすることになります．少々難しい話ですが，一応ストーリとして把握してください．

多項式の因数分解を係数から直接行うのはとても難しいので，代わりに極（pole）と零点（zero）を使います．極と零点はフィルタの特性を示す重要なパラメータですが，極と零点とは，伝達関数の分子と分母それぞれの多項式の根（多項式が0となる点の座標）なのです．多項式の根が分かれば因数分解することは簡単です．

ではどうやって極と零点を求めるかというと，実はSciPyのフィルタ設計関数にその機能が備わっています．先ほどは伝達関数を多項式の係数列b，aとして得ました．これを*ba*形式と呼ぶことにします．*ba*形式の代わりに，極と零点の集まり（これを*zpk*形式＝*zero*, *pole*，そして係数*k*と呼ぶ）として得ることが可能です．scipy.signal.ellip関数に，オプションとしてoutput='zpk'というパラメータを与えます．6次のフィルタであれば，零点が6個，極が6個，係数が一つ得られます．実行した結果を**図8**に示します．

六つの零点はすべて複素数ですが，複素共役（実部が同じで虚部の符号が反対）なペアが三つ得られます．このペア一つが2次多項式に対応しています．ペアが三つあるので，2次多項式が三つに相当します．同じく極についても3ペア得られますので，これも三つの2次多項式に相当します．この極のペアと零点のペアを一つずつ組み合わせると2次フィルタを構成できます．全部で三つの2次フィルタができあがります．

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_n z^{-n}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n}}$$

図6　IIRフィルタの伝達関数
分子と分母がそれぞれ*N*次の多項式として表現できる

```
In [3]: w,h = signal.freqz(b, a)
        w = w * 8000 / (2 *np.pi)
        pl.ylim((-70, 10))
        pl.grid()
        pl.plot(w, 20 * np.log10(abs(h)));
```

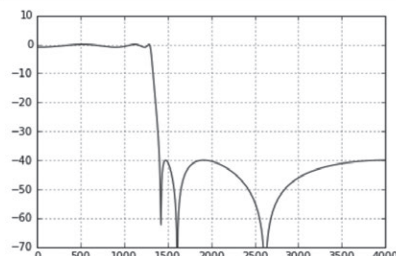


図7　設計した楕円ローパス・フィルタの周波数特性

```
In [4]: z, p, k = signal.ellip(6, 1, 40, 1300.0/4000, 'low', output='zpk')
        z,p,k

Out[4]: (array([-0.46711900+0.88419446j,  0.30230927+0.9532099j ,
                 0.43977552+0.89810773j, -0.46711900-0.88419446j,
                 0.30230927-0.9532099j ,  0.43977552-0.89810773j]),
          array([ 0.63729472-0.31957954j,  0.54848834-0.71171772j,
                 0.51231440-0.83532607j,  0.63729472+0.31957954j,
                 0.54848834+0.71171772j,  0.51231440+0.83532607j]),
          0.030175331934605188)
```

図8　フィルタ設計関数にパラメータを指定すると係数などが得られる

出力を*zpk*（ゼロと極）形式とすることで，ゼロ点と極がそれぞれ六つ配列として得られた．ゼロ点や極は複素数で，虚数部の符号が違う複素共役がそれぞれ3組ある．最後の数値kは全体の係数

この2次フィルタの特性を確認してみましょう．zpk2tfという関数を使うと，*zpk*形式から*ba*形式に変換できます．そしてfreqz関数を使うと周波数特性が得られます．これを一つのグラフに描画したものが**図9**です．一つ一つはとがったピークと落ち込みのある特性ですが，全部を乗じて合わせると，最初に求めたローパス・フィルタの特性が得られます．

▶カスケード構成フィルタ設計の注意点

注意が必要なのは，フィルタを順番に適用する際に，飽和を生じないようにすることです．周波数特性が1を超えてしまうと，そこで飽和する可能性があります．そのため，1を越えるピークのあるフィルタは，前段で減衰した箇所になるよう順番を検討します．

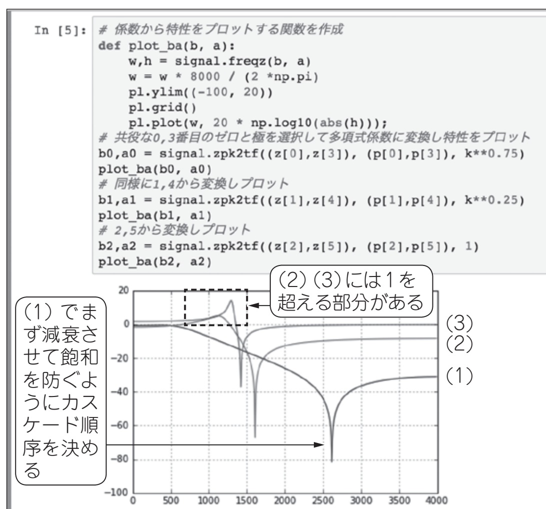


図9　カスケード構成フィルタ設計の注意点…三つの2次フィルタに分解してゲインを分配し，それぞれ飽和しないようにする

また，全体にかかる係数kは各段に分配するのですが，その分配についても飽和が発生しないように適切に行うようにします．係数kの分配した各段の特性をグラフにして検討します．**図9**に示す(1)，(2)，(3)の順序でフィルタを適用することが適当だろうと判断しました．**図10**に伝達関数の係数を示します．

設計したフィルタ係数をマイコン用のソースコードに転記する

三つの2次フィルタの係数が得られたので，これをCMSIS-DSPに用意されているarm_biquad_cascade_df1_q15関数で使えるよう係数を整えます．フィルタに与える係数は符号付き16ビット固定小数点(q15)で表現する必要があります．設計で得られた係数は1を越えるものがありますが，q15形式では1を越える数値を取り扱うことができません．そのため16ビットに納まるよう1ビット右シフトした数値として扱うため32768>>1=16384を乗じたうえで，数値を整数に丸めます．結果を**図11**に示します．

この得られた数値を，arm_biquad_cascade_df1_q15関数の仕様[3]に従って並べます．scipy.signalの*ba*形式とは*a*(分子の多項式の係数)の符号が反対なようなので，それに注意してソースコードに転記します．フィルタを使用するための必要な係数と，状態変数などの記述例を**リスト1**に示します．

以上，Jupyter NotebookとPythonによるフィルタ設計を紹介しました．パラメータをトライ・アンド・エラーで変えて特性グラフを確認しながら設計を進めることができます．ファイルとして保存できるので，そのまま設計資料にしたり，githubやgistで公開したり[4]できます．

```
In [6]: print (b0, a0)
        print (b1, a1)
        print (b2, a2)

        [ 0.07240008  0.06763891  0.07240008] [ 1.         -1.27458944  0.50827564]
        [ 0.4167859  -0.25199648  0.4167859 ] [ 1.         -1.09697667  0.80738157]
        [ 1.         -0.87955103  1.        ] [ 1.         -1.02462881  0.9602357 ]
```

図10　三つすべての2次フィルタの係数を確認してみる

```
In [7]: # 固定小数点とするため16384を乗じて、np.rintで整数に丸める
        print (np.rint(b0*16384), np.rint(a0*16384))
        print (np.rint(b1*16384), np.rint(a1*16384))
        print (np.rint(b2*16384), np.rint(a2*16384))

        [ 1186.  1108.  1186.] [ 16384. -20883.   8328.]
        [ 6829. -4129.  6829.] [ 16384. -17973.  13228.]
        [ 16384. -14411.  16384.] [ 16384. -16788.  15733.]
```

図11　16ビット固定小数で取り扱うため整数化して丸めたフィルタ係数

16ビット固定小数点で－1～1に相当する－32768～32767に収まるように1ビット右シフトしてある

リスト1　フィルタ係数をソースに転記する

必要な変数とエリアを用意しフィルタ関数を呼び出す

```
// フィルタの状態を保存するための配列（3段分）
q15_t bq_i_state[4 * 3];
q15_t bq_q_state[4 * 3];

// 2次IIRフィルタ3段カスケードの係数
q15_t bq_coeffs[] = {
       1186, 0,   1108,  1186, 20883,  -8328,
       6829, 0,  -4129,  6829, 17973, -13228,
      16384, 0, -14411, 16384, 16788, -15733
};

// 同じ係数のフィルタを二つ（i,q）使用する
// 3段、フィルタ状態配列、フィルタ係数、1ビットシフト
arm_biquad_casd_df1_inst_q15 bq_i = {
                      3, bq_i_state, bq_coeffs, 1};
arm_biquad_casd_df1_inst_q15 bq_q = {
                      3, bq_q_state, bq_coeffs, 1};

void demod() {
 ...
    // input_i,qの信号をフィルタしてoutput_i,qに出力
    arm_biquad_cascade_df1_q15(&bq_i, input_i,
                                      output_i, size);
    arm_biquad_cascade_df1_q15(&bq_q, input_q,
                                      output_q, size);
 ...
}
```


◆参考・引用＊文献◆

(1) 高橋 知宏，林 輝彦；特集「初体験！ オール・ソフトウェア無線」，Interface，2015年7月号，CQ出版社．
(2)＊高橋 知宏；7MHz帯SSB受信機レベルアップ②！ 自動ゲイン調整の追加，高速ワンチップ・マイコンではじめるソフトウェア無線 第8回，Interface，2016年5月号，CQ出版社．
(3) CMSIS DSP Software Libraryリファレンス．
(4) 本稿関連の公開ページ．
https://gist.github.com/ef88c3709c5725c2637f

たかはし・ともひろ

コラム　Pythonを仕事で使うにはライセンスの調査が必要　佐藤 聖

● まずはPSFL契約内容を確認する

Pythonと標準ライブラリ（一部を除く）は，Python Software Foundation License（PSFL）です．ソースコードをオープンにしなくとも，オリジナルのソースコードの修正が許されており，Pythonプロジェクトの配布，商用利用も可能です．

ただし，ソースコードの書き換え，ほかのソースコードへコピー，静的リンク，動的リンクが発生する場合には，PSFLの契約および著作権表示が必要ですので，事前にPSFL契約内容を確認する必要があります．

● ライブラリが呼び出しているライブラリの権利も気にしなければならない

Pythonのサード・パーティ・ライブラリも豊富にあります．大抵のアルゴリズムは一から開発しなくても，公開ライブラリを利用することで開発の短期間化・省力化が可能です．しかし，ライブラリがライブラリを参照するような形を取っていることが多々あり，複数のライセンス契約（BSD，MIT，Apache，LGPLなど）に従わなければなりません．

ライセンスの概要は，GNUサイト掲載の「さまざまなライセンスとそれらについての解説」（https://www.gnu.org/licenses/license-list.ja.html）が参考になります．また，ライブラリのバージョン・アップに伴い，ライセンスの種類が変更されることもあります．仕事ではPythonよりも，ライセンス契約が明確なC#やJAVAがよく利用されます．

● ライセンス違反で係争に発展することも

各種ライセンスの被許諾者は契約内容を理解し，適切な対応が求められます．その上で，各国の知的財産権関連法や輸出管理法などに注意を払わなければなりません．

米国や欧米では，オープンソース・ライセンスの違反が係争に発展することがしばしばあります．

また，クラウド・サービスを提供するケースでも，海外のサーバを利用する際には注意が必要です．Pythonのパッケージに含まれるOpenSSLなどのセキュリティ技術を国外に輸出する場合，監督官庁への届け出が必要になるかもしれません．

このような管理の複雑さが，仕事でPythonを利用しない要因の一つになっています．

第6章 OpenCVよりビギナ向け！画像操作ライブラリPillow

ダウンロード・データあります

Pythonで数学の描画問題を解く

岩城 信二

Pythonは，科学技術計算だけでなく，画像処理・描画を行えるライブラリもいろいろあります．本稿では，その中でも定番OpenCVよりビギナ向けで，単純な画像操作が行えるライブラリPillowを使って，数学の一筆書き問題を解いて画像の描画を行ってみます．（編集部）

Pythonの画像ライブラリ

● 種類

表1に示すように，Pythonには多くの画像ライブラリが存在しますが，広く使われているのはPillowとOpenCV-Pythonの2種類です．ライブラリごとに提供するメソッドが異なったり，速度に差があったりしますが，基本的にはPillow, OpenCV-Pythonのどちらかを使えば大抵の処理には対応できること，ウェブ上のコミュニティも大きいことから，この二つのどちらかを学習することをお勧めします．

単純な操作（拡大/縮小/描画など）はPillowを，高度な処理にはOpenCV-Pythonを利用するといった使い分けをするプログラマが多いです．

OpenCVの方が機能が豊富であり，OpenCV-Pythonさえ覚えればPillowは不要という考え方もあります．今回は以下の理由からPillowを取り上げます．

● 画像ライブラリPillowのメリット

▶（1）OpenCV-Pythonと比べてPillowはインストールが簡単

```
pip install pillow
```

でインストール可能です．また，科学技術計算用ディストリビューションのAnacondaにも含まれています．

▶（2）OpenCV-Pythonと比べて記述が楽

OpenCV-Pythonは記述に一癖あり，初学者にとっては障壁が高くなっています．例えば，画像をグレースケールに変換する際の記述を見れば一目瞭然です．

表1　Pythonの主な画像ライブラリ

	ライブラリ名	特　徴
ビギナ向け	PIL (Python Imaging Library)	各種形式の画像ファイルの読み込み/操作/保存を行う機能を提供するフリー・ライセンス・ライブラリ．Pythonで画像処理といえばOpenCV-Pythonと並んでPILを用いるケースが多いが，2011年6月28日でソース改修がストップしている．PILの後継として，Pillowという後継プロジェクトがPILのリポジトリをフォークして進められている
ビギナ向け	Pillow	PILの後継ライブラリで，現在も活発にメンテナンスされている．品質/機能共にPillowがPILに比べて高いため，Pillowが使わている．Pillowのモジュール名もPILであるため，上記のPILとは混在できないことに注意
定番	OpenCV-Python	OpenCV (Open Source Computer Vision Library) と呼ばれるインテルが開発した画像処理ライブラリのPython用ライブラリ．Pillowは画像をイメージ・オブジェクトとして取り扱うが，こちらはNumPyのndarrayとして画像を取り扱うため，SciPyやscikit-learnといったその他ライブラリとの連携が手軽に行える
	scikit-image	OpenCV-Pythonと同様，画像をNumPyのndarrayとして取り扱うため，その他ライブラリとの連携が手軽に行える．OpenCV-Pythonとの使い分けは難しいが，一部OpenCV-Pythonにない手法があったり，特定の手法に限ってはsickit-imageの方が高速といった点で使い分けが行われている
	SciPy	プログラミング数学，科学，工学のための数値解析ライブラリ．画像処理機能も含まれており，こちらもscikit-image同様，OpenCV-Pythonと使い分けされている
	Mahotas	C++で実装された高速な画像処理ライブラリ．こちらもNumPyのndarrayとして画像を取り扱うことが可能
	PythonMagick	ImageMagickという画像処理用のフリー・ソフトウェアのPythonインターフェース・ライブラリ
	Pycario	2D画像ライブラリcarioのPython用ライブラリ
	SimpleITK	Insight Segmentation and Restration Toolkit (ITK) と呼ばれるライブラリで，画像レジストレーションと呼ばれる技術の手法が豊富なライブラリ

表2　画像の読み込み/作成時におけるPillowライブラリの機能

機　能	説　明
画像の読み込み	［例］img = Image.open("sample.jpg") ・引き数には読み込みたい画像ファイルを指定する ・bmp, tiff, jpg, pngなどのメジャーなフォーマットに対応する
画像の作成	［例］img = Image.new('RGB', (300,300), (255, 255, 255)) ・第1引き数は作成したい画像のモードを指定する （透過度も持たせたい場合は'RGBA'とする．白黒の場合は'L'） ・第2引き数は画像サイズ（幅，高さの順）を指定する ・第3引き数はデフォルトで用いられる色の指定する
イメージ・オブジェクトの属性	上記で作成したイメージ・オブジェクト（img）は，以下の属性を持つ ・img.size：幅，高さ（width, height）のタプル（tuple） ・img.format：読み込んだファイルの拡張子

• OpenCV-Pythonの場合

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

• Pillowの場合

```
gray_im = im.convert('L')
```

Pillowを使った応用としては，ウェブ・ページに載せる画像を適切なサイズや拡張子に変更する際に，大量の画像を一括変更することが可能です．他の画像や文字列を合成することができ，画像にロゴや文字列を透かしとして入れる際に，この処理を一括実行することも可能です．また，特定の物体（例えば赤い板）の座標を検知することができ，赤い板を持たせた人物を撮り，その中に時刻（文字列）を合成すればオリジナル美人時計も作成可能です（http://www.bijint.com/）．

● 画像ライブラリPillowの基本機能

ここでは，Pillowの基本的な機能の使い方を記述します．以下の流れに沿って説明します．

(1) 既存画像の読み込みもしくは新規画像の作成
(2) 画像処理（拡大，縮小，変換，切り出し，統計情報取得）
(3) 画像描画
(4) 保存

なお，詳細については公式チュートリアルhttp://pillow.readthedocs.io/en/3.1.x/handbookを参照してください．

▶機能（1）：既存画像の読み込みもしくは新規画像の作成

画像の読み込み/作成/イメージ・オブジェクトの属性について**表2**に示します．画像の読み込み/作成はImageクラスを用います．

```
from PIL import Image
```

でインポートできます．

▶機能（2）：画像処理

作成したimgオブジェクトのメソッドを用いて各種操作を行います．**表3**に示すように画像処理の機能には，画像の拡大/縮小/回転/切り出し/色変換，ピクセルの取得/変更，画像の統計情報取得などがあります．

▶機能（3）：画像描画

画像の描画は，ImageDrawモジュールを用います．from PIL import ImageDrawでインポートします．

表4に示すように画像の描画機能には，Drawオブジェクトの生成や，任意の2点間に線を引いたり，任意の点に円を描いたりします．

▶機能（4）：保存

画像の保存は，イメージ・オブジェクトのsaveメソッドを用います．以下のように，保存したい形式の拡張子付きで保存先を指定します．

```
img.save("sample_resized.jpg")
```

今回解いてみる数学…レガシー一筆書き問題TSP

● 任意の点群を一巡する（一筆書き）総移動コストを最小にする

TSP（Traveling Salesman Problem）は巡回セールスマン問題と呼ばれる組み合わせ最適化問題です．任意の点群を一巡するのが最短となる経路を求めます．

TSPはオペレーションズ・リサーチや理論計算機科学の分野で長きに渡って研究されてきた問題です．「地点の集合と2地点間の移動コストが与えられたとき，全ての地点をちょうど1度ずつ巡り出発地に戻る巡回路の総移動コストが最小のものを求める」組み合わせ問題として定義されています．

図1には実際の問題の例を示しています．**図1(a)**に100個の点が存在しており，これら全てを通る最短の巡回路（解）が**図1(b)**に示されています．

一見，簡単そうに見えますが，地点が増えれば増えるほど組み合わせ数が爆発的に増え，解くのが極端に難しくなります．事実，TSPは効率的に（多項式時間内に）最適解を見つける方法がいまだに発見されていません．

ただし，TSPでは最適解ではないにせよ，精度の

表3 画像処理におけるPillowライブラリの機能

機 能	説 明
画像の拡大/縮小	[例] resize = img.resize((100,100)) ・第1引き数はリサイズ後のサイズを幅，高さのタプルで指定する ・上記の例の場合，画像を100×100ピクセルにリサイズする
画像の回転	[例] rotate = img.rotate(90) ・引き数は回転させたい角度を指定する(時計回り) ・上記例の場合，右に90°回転している
画像の切り出し	[例] crop = img.crop((10,0,20,20)) ・引き数は画像の中から抜き出したい領域を指定(左，上，右，下の4数値のタプルで指定)する ・Pillowの画像オブジェクトの座標系は，x軸は画像左端が0，y軸は画像上端が0になる
画像の色変換	[例] gray_img = img.convert('L') ・引き数は変換モードを指定する('L'は白黒を意味する) ・その他のモードについては公式チュートリアルを参照.
ピクセルの取得	[例] pix = img.getpixel((10,10)) ・引き数はx，y座標のタプル. ・上記の場合，x=10，y=10のピクセル色情報がpixに格納される
ピクセルの変更	[例]：img.putpixel((10, 10),(10, 10, 10)) ・第1引き数はx，y座標のタプル ・第2引き数は変更後の色を指定する('RGB'の場合は('R','G','B')のタプルで，白黒の場合は数値を入れる) ・上記の場合，x=10，y=10のピクセルの色をRGB=(10, 10, 10)に変更している
画像の統計情報取得	from PIL import ImageStatでImageStatモジュールをインポートする [例] stat = ImageStat.Stat(img) ・引き数は画像オブジェクトを指定する 作成されたstatオブジェクトは以下の属性を持つ(画像モードによって出力が変わる．ここでは'RGB'として画像を読み込んだと想定して説明する) ・extrema：各バンド('R'，'G'，'B'それぞれ)の最大値・最小値を返す [例][(20, 200),(10, 232),(100, 123)] ・mean：各バンド('R'，'G'，'B'それぞれ)の平均値を返す ・stddev：各バンド('R'，'G'，'B'それぞれ)の標準偏差を返す

高い解を算出する方法が数多く考案されてきました．

なお，TSPについてもっと詳しく知りたい方は参考文献(1)を参照ください．

● 実際いろいろ使える…一筆書き問題の応用

TSP(巡回セールスマン問題)という名称からして，この問題の研究が応用ありきで進められてきたことは明らかかと思います．以下に，ほんの一部ではありますが，TSPがどのような問題に応用されているか紹介します．

▶ (1) 配送計画への応用

バスやトラックが人や荷物を載せ，複数の目的地に届ける道を決めるときに，TSPモデルが多く使われています．総走行距離を短くすることにより，燃費や人件費の削減を実現できます．

▶ (2) 産業用機械への応用

プリント基板のドリル穴あけ，はんだ付けといった加工を行う産業用機械に，TSPモデルが多く使われています．基板には数百から数千個の加工が必要な点があり，経路が最短となるような加工順をTSPを応用して求めることにより，消費電力の削減や製造時間の短縮を実現できます．

▶ (3) 惑星探索

NASAによる宇宙望遠鏡用の計画作業．望遠鏡の向きを変更するには大量の燃料を要するため，なるべく変更量が少なくするような観測巡回路をTSPを用いて算出しています．

▶ (4) その他

倉庫から商品を収集する際の巡回路探索や，ガラス工場での切り出しパターンの算出，また最近話題になっているPokémon GOへの応用として周囲のポケストップとジムを最短で回る巡回路の算出などがあげられます．

● TSPの一筆書きを「絵」ととらえてみる…TSP Art

任意の写真を点で描画したTSPを題材としたアート作品をTSP Artといいます．実際の例を図2に示します．元の画像が左端，中央が点描画したもの，右端が完成形(TSP Art)です．

本稿では，見てイメージがわきやすいように，TSP Artとして描画問題を解いてみます．

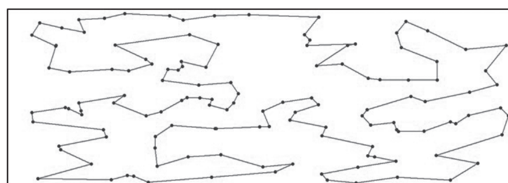
TSP Artは，最適な巡回路が必要とされるわけではないので，今回は精度の高い解を算出する「近似」解法を用いることになります．

表4　画像描画におけるPillowライブラリの機能

機　能	説　明
Drawオブジェクトの生成	［例］draw = ImageDraw.Draw(img) ・引き数は描画を行いたいイメージ・オブジェクトを指定する
任意の2点間に線を引く	［例］draw.line (((0,0),(10,10)), fill=(127, 127, 127), width=2) ・引き数は始点と終点を指定する ・上記の例は点(x=0, y=0)から点(x=10, y=10)に線を引いている ・fillは色を，widthは線幅を指定する
任意の点に円を描く	［例］draw.ellipse([(9,11),(11,9)],fill=(127,127,127) ・引き数は描きたい楕円の左上点と右下点を指定する ・上記の例は中心点(x=10, y=10)に直径2の円を描いている．左上点(9, 11)，右下点(11, 9)を指定する ・fillは円内部の色を指定する



(a)点在した100個の点



(b)点を通る巡回路(解)

図1　レガシな数学描画問題TSP…任意の点群を一巡する(一筆書きする)総移動コストを最小にしたい

実験の準備

● 実行環境

Linux，UNIX，Macなどで動作しますが，今回の作業環境は以下の通りです．

OS：Windows 10(64ビット)
Pythonバージョン：3.4.3

● 必須ライブラリ

Pythonは科学計算ライブラリが充実しており，読み込み，計算，描画を一貫して行えることが大きな利点です．以下4種類のライブラリを用います．Anacondaをインストールした方は，以下のライブラリは既に含まれています．

(1) Pillow(PIL)

(2) SciPy

プログラミング数学，科学，工学のための数値解析ライブラリです．今回使用するボロノイ図の作成ライブラリなど，数多くの数値解析アルゴリズムを提供します．Rubyなどのプログラミング言語を使っていた科学者がPythonに移行した大きな理由が，SciPyであったと言われています．

(3) NumPy

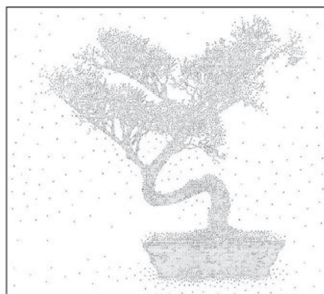
数値計算を効率的に行うためのライブラリです．また高水準の数学関数ライブラリも提供します．

(4) matplotlib

グラフ描画ライブラリです．取り扱いが非常に簡



(a)元の画像



(b)点描画した画像



(c)完成形(TSP Art)

図2　今回解く数学問題TSP…任意の点を最短で一筆書きする経路を求めて描画してみる
任意の点群を写真から生成して描画した一筆書きの絵をTSP artという

リスト1　TSP ARTのメイン・プログラム(tsp_art.py)

```
from PIL import Image
from PIL import ImageDraw
from stipple import stipple
from tsp import nn,opt2

def draw_route(route, sz, fname, red=set(),
green=set(), blue=set()):
    im = Image.new('RGB', sz, (255, 255, 255))
    draw = ImageDraw.Draw(im)
    edges = [[route[i],route[i+1]] for i in
range(len(route)-1)] + [(route[-1], route[0])]
    for e in edges:
        draw.line(e, fill=(127, 127, 127), width=2)
    del draw
    im.save(fname)

# 画像の読み込み
im = Image.open('bonsai.jpg')
im = im.resize((int(im.size[0]/1.5),int(im.
size[1]/1.5)))
gray_im = im.convert('L')
gray_im.save('gray_converted.jpg')
print("横:{}px, 縦:{}px".format(im.size[0],im.
size[1]))

# 画像を点描画
points = stipple(gray_im,5)

# 点群をTSPとして解く
nn_route = nn(points)
draw_route(nn_route, im.size, 'tsp_nn.jpg')
route_improved = opt2(nn_route)

# 経路描画
draw_route(route_improved, im.size, 'tsp_improved.
jpg')
```

単かつ高機能なため，Pythonでグラフを描画する際によく使われています．簡単な幾何計算機能も保持しており，今回のプログラムではグラフ描画ではなく幾何計算機能を使うために本ライブラリを用いています．

ソースコード

本稿で紹介するソースコードは，以下の筆者のダウンロード・ページから入手可能です．

https://github.com/shinji071/tsp_art

下記の3種類のソースコードを用意しました．

(1) tsp_art.py：メイン・プログラム
(2) stipple.py：点描画用プログラム
(3) tsp.py：TSPを解くためのプログラム

リスト1にメイン・プログラム(tsp_art.py)を示します．今回のプログラムの構成を**図3**に示します．

stipple()，nn()，opt2()のメソッドに関しては，stipple.py，tsp.pyに含まれています．TSP Art作成には，以下四つのステップを経る必要があります．

(1) 画像の読み込み

任意の画像を読み込み，点描画するために白黒画像に変換します．

(2) 読み込んだ画像を点描画

(1)で読み込んだ画像を点だけで表現します．Weighted Voronoi Diagramという手法を利用します．

(3) (1)で生成された点群をTSPとして解く

(2)で生成した点群をTSP問題として捉え，解を算出します．今回は最適解である必要はないため，Nearest Neighborhood法と，2-opt法を組み合わせた方法で解を算出します．

(4) (3)で導かれた経路を描画

(3)で得られた巡回路を描画し，画像として出力します．

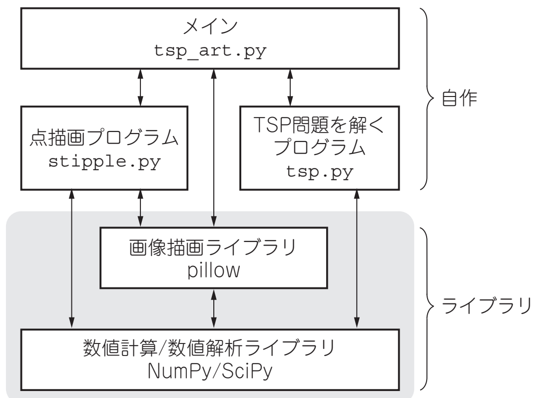


図3　一筆書き問題TSPを解いて描画するためのソフトウェア構成
任意点群を写真から生成したTSP Artとして描画させる

● ステップ1：画像の読み込み

Pillowの基本的な使い方で説明した通りに読み込みます．画像が大きい場合は，resizeメソッドを用いてサイズを調整します．サイズが大きすぎると計算に時間がかかるため，大きくても1000ピクセル×1000ピ

表5　自作点描画stippleメソッドの引き数および返り値

メソッド	説　明
引き数	第1引き数：グレースケールのImage(gray_im) 第2引き数：Weight Voronoi Diagram作成にあたっての繰り返し回数．大きい値であればあるほど精度が高くなるが，計算時間が長くなる．画像サイズにもよるが，5程度で十分にきれいな点描画ができ上がる
返り値	点(x, y)タプルのリスト [例] [(1, 2),(6, 3),(3, 4), …, (10, 19)]

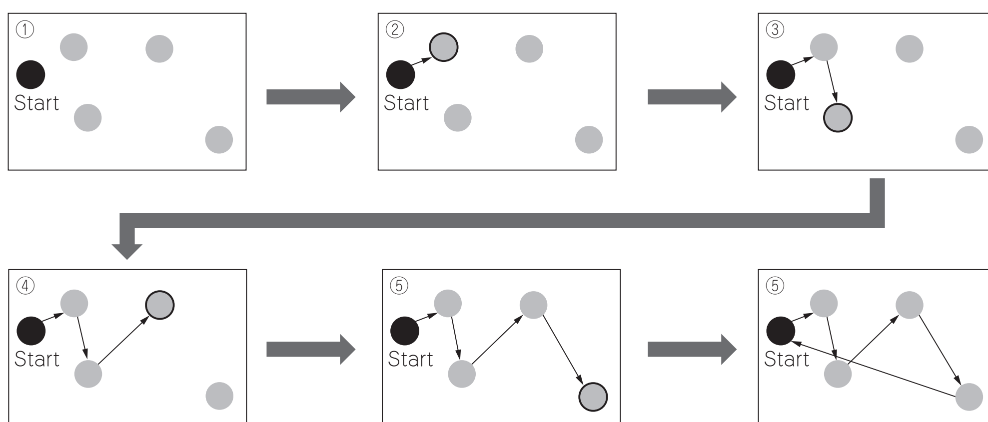


図5　ある点から始めて一番近い点に線を引いていくという単純な方法(Nearest Neighborhood法)

クセル程度とします(サンプル画像のサイズ853ピクセル×770ピクセルの場合で総計算時間は30分程度).

最後に画像をグレースケール化しますが，convertメソッドを使います.引き数に“L”を渡すことにより，画像がグレースケール化します.

```
im = Image.open('sample.jpg')
im = im.resize((int(im.size[0]/1.5),int(im.size[1]/1.5)))
gray_im = im.convert('L')
```

● ステップ2：画像を点描画

画像の点描画はstippleメソッドを用います. stippleメソッドのソースコードは`stipple.py`に含まれています．ソースコードは誌面の都合でここでは割愛しますが，ダウンロード・データとして提供します．処理の詳細に興味がある場合は参照できます．

点描画はグレースケール写真の黒いところでは密に，白い箇所には疎に配置され，それらの間隔が均一であることが望ましいと言われています．ここでは，参考文献(2)に記した論文のアルゴリズム(Weight Voronoi Diagramの生成)をベースにプログラムを作成しました．stippleメソッドの引き数および返り値について**表5**に示します．得られた点タプルを描画すると，**図4**のようになります.



図4　点描画した画像

● ステップ3：点群をTSPとして解く

TSPはNP困難と呼ばれる問題群に属しており，最適解を効率的に求める方法がありません．従ってTSP Artの観点では，線が交差しない巡回路を算出できれば十分なので，以下の近似解法を用います.

▶ (1) Nearest Neighborhood法(nn)

Nearest Neighborhood法は，ある点から始めて一番近い点に線を引いていくという単純な方法です．本方法のイメージを**図5**に示しました．以下のメソッドで実行できます.

```
route_nn = nn(points)
```

nnメソッドは`tsp.py`に含まれています．`points`を引き数に渡すことにより，Nearest Neighborhood法で解かれた解が`route_nn`に格納されます.



図6　TSPアルゴリズム①…Nearest Neighborhood法で算出した巡回路
線が交差している．見栄えも美しくない

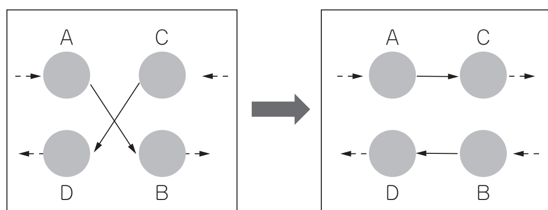


図7　任意の2本の辺をつなぎ直して新しい巡回路を作る方法（2-opt法）
辺の長さ「AB＋CD ＞ AC＋BD」のとき，ABとCDをACとBDへ交換する

route_nnの中身は点（*x*，*y*）タプルのリストとなっています．

route_nnの先頭から末尾までの点をなぞって描画すると，Nearest Neighborhood法で算出した巡回路を描画できます．

ただし，Nearest Neighborhood法で算出した解は，精度が悪いことに加え，線が交差しているなど，**図6**に示すように見栄えが美しくありません．そこで，得られた解を改良すべく，以下の2-opt法を適用します．

▶ **(2) 2-opt法**

2-opt法は，すでに出来上がった巡回路（今回の場合はroute_nn）に対して，任意の2本の辺をつなぎ直して新しい巡回路を作るという方法です．

図7のように，辺ABと辺CDに着目した場合，辺の長さは「AB＋CD ＞ AC＋BD」であり，辺ACと辺BDの方が明らかに短いことが分かります．従って，辺ABおよびCDを取り除き，新たに辺ACと辺BDを追加します（辺AB，CDを辺AC，BDと交換する）．この処理を巡回路にある全ての辺のペアに適用していき，交換するペアがなくなった時点で，2-opt法は完了になります．

以下のメソッドで実行できます．

```
route_improved = opt2(nn_route)
```

opt2メソッドはtsp.pyに含まれています．ソースコードは誌面の都合で割愛しますが，ダウンロード・データとして提供します．処理の詳細に興味がある場合は参照できます．route_improvedの中身は点（*x*，*y*）タプルのリストとなります．

2-opt法の実行により，解精度が向上するのはもちろんのこと，**図8**に示すように線が交差しないといった見栄えも改善できます．参考文献(3)に2-optの様子が可視化された動画を見ることができます．

● ステップ4：経路描画

最後に，3で得られた結果をPILのImageDrawクラスを用いて描画し保存します．

以下のメソッドで行っています．



図8　TSPアルゴリズム②…2-opt法で算出した巡回路は交差もなくてシンプル

```
draw_route(route_improved, im.size,
'tsp_improved.jpg')
```

上記メソッドは，tsp_art.py内に記載されており，Pillowを用いて描画を行っています．

＊　＊　＊

TSP Artをきれいに本格的に作りたい場合，解像度の高い画像が必要となると同時に，点描画作成時のパラメータを調整する必要が出てきます．Pythonに詳しい方は本プログラムのパラメータを諸々調整するのも一つの手ですが，参考文献(4)のサイトの方法を試すのも一つの手です．このサイトでは，プログラムではなく，既存のフリー・ソフトウェアを用いてTSP Artを作る方法を紹介しています．

また，Jupyter Notebookと呼ばれるウェブ・ブラウザ上で起動する動作環境を用いれば，途中結果を確認しつつ効率的に作業を進めることもできます．興味のある方は参考文献(5)を参考にしてください．

◆参考文献◆

(1) William J. Cook，松浦 俊輔 訳；驚きの数学 巡回セールスマン問題，青土社，2013年.
(2) A. Secord；Weighted Voronoi stippling，Proc. 2nd Int. Symp. Non-Photorealistic Animation and Rendering（NPAR），pp.37-43，2002年.
https://www.cs.ubc.ca/labs/imager/tr/2002/secord2002b/secord.2002b.pdf
(3) https://www.youtube.com/watch?v=SC5CX8drAtU
(4) http://drububu.com/illustration/tsp/index.html
(5) Cyrille Rossant，菊池 彰 訳；Pythonデータサイエンスクックブック ―対話型コンピューティングと可視化のためのレシピ集，オライリージャパン，2015年.

いわき・しんじ

第7章 1500円定番ARMマイコン・ボードで試して合点！

アプリはスクリプトで柔軟に！マイコン用MicroPythonプログラミング

中村 晋一郎

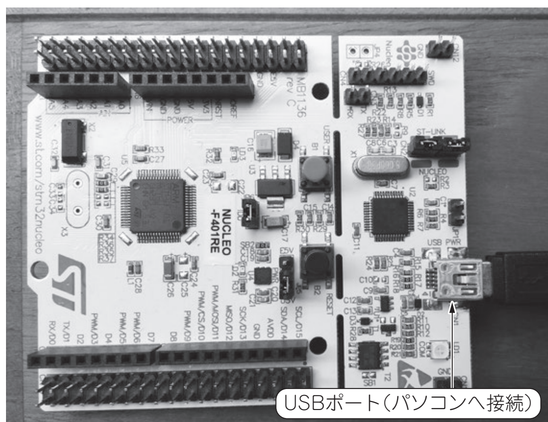


写真1 今回やること…マイコン用PythonスクリプトMicroPythonを動かしてマイコンCプログラム（ハードウェア依存部）**を一切いじらなくても上位機能を記述できるようになることを確認する**
実験ボードNUCLEO-F401RE．秋月電子通商で1,500円（2016年7月現在）で買える

本稿では，データ解析などに長けた世界の定番スクリプト言語Pythonのマイコン版であるMicroPythonを動かせるようにして，マイコンCプログラムを一切いじらなくても上位機能を記述できるようになることをマイコン・ボード（NUCLEO-F401RE）で体験してみます（**写真1**）．（編集部）

マイコン用Pythonスクリプト MicroPythonのススメ

● スクリプト言語をマイコン・プログラムに組み込むメリット

古典的な組み込みシステムの場合，全ての論理がコンパイルの必要な言語で記述されており，後から変更する場合にはコンパイルと再インストールが必要でした．

システムの変更に柔軟に対応することが難しく，対応できたとしても保守の難しいシステムになってしまうことも少なくありません．

近年，システム下層機能をCやC++でがっちりと実現しておきながら，上層機能をスクリプト言語で実現する，いわばハイブリッドな構成が注目されるようになっています．

このような構成をとることで，システムの中心的な機能は従来通りCやC++で実現しつつも，上層機能の柔軟性をスクリプト言語で確保できます．古典的な組み込みシステムでは得られなかった機能の柔軟性と堅牢性を両立できる設計手法です．

● コア機能とアプリケーション機能の分離にスクリプトを利用する

本稿では，システム下層機能をそれ以上分割の難しい機能単位で区切るものとしてプリミティブな機能，上層の機能をサービスと便宜上呼ぶこととします．

システムが持っている「プリミティブな機能」は，ハードウェア，ミドルウェアなどの外部制約から実現可能な機能はおのずと決まっており，比較的変更が少ない部分，あるいは外部制約上変更が難しい部分です．

ユーザに最も近い「サービス」の部分は，一般に「プリミティブな機能」を組み合わせて実現されます．ユーザが触れる部分は，一般に変更の要求が入りやすい箇所といえ，ユーザによっては異なる処理が望まれるかもしれません．コンパイルの必要な言語で実装されている場合には，設計者が動作を変更してコンパイルする必要があります．

スクリプト言語で実装されていれば，スクリプト言語の部分のみを変更すれば動作を変えられます．スクリプト言語を搭載することで変更要件の入りやすい箇所を柔らかく実現しておけるので変更を受け入れやすく，結果的にさまざまな発展の可能性を将来に残しておけます．

● マイコン用Pythonスクリプト「MicroPython」誕生の背景

MicroPythonは，マイコン上で実行できるように最適化されたPythonで，Damien P. George氏によって始められたプロジェクトです．

PyCon UK 2014で発表されてから徐々に有名になり，今ではマイコン上でPythonを動作させたい場合のデファクト・スタンダードとも言える状態になりつ

つあります．

もともとこのプロジェクトは，Pythonのようなスクリプト言語で操作できたら開発を高速化できるだろうというプロトタイピングに対する興味と，実際にマイコン上でPythonのような言語を実行可能にできるのかという技術的な興味が相まって始まったプロジェクトのようです．

● MicroPythonの特徴

さまざまなマイコン上で動かせるように，さまざまな工夫がされており，すでに多数のマイコンでの動作実績があります．例えば，オブジェクトの定義はマクロで簡単にできるようになっており，特に難しいことを考えなくとも既存の実装を参考に追加することができます．これは，後ほど説明する追加手順を体験いただくことでも納得いただけるかと思います．

MicroPythonのリポジトリを見ると今回紹介するSTM32のほかに，`esp8266`，`cc3200`，`pic16bit`などのディレクトリがあります．各ディレクトリには環境に応じたMakefileやプロジェクト・ファイル，ポートによっては説明を書いたファイルもありますので，興味のある方は見てみてください．

● Pythonがマイコンで動くことのメリット

世の中にはさまざまなスクリプト言語があります．マイコン向けに使えるスクリプト言語には以下のようなものがあります．

(1) Luaに対するeLua
(2) Rubyに対するmruby
(3) Pythonに対するMicroPython

システムに要求する最小要件はだいたい似たり寄ったりでROMは128Kバイト前後から，RAMは8K～32Kバイトくらいからを推奨条件としているようです．

しかし，例えばLuaには言語の機能としてクラスが存在しません．これは意外に面倒なときがあるかもしれません．MicroPythonは，オブジェクト指向言語としての機能をサポートしています．加えて独自の機能を追加することを念頭に設計されており，後述する新しい機能への対応も簡単にできるようになっています．

システムのコアとなる機能はCやC++など従来の組み込みシステムで用いられる言語で記述し，アプリケーション層に近い部分をMicroPythonで記述することで，変更要件に応じて「スクリプトを変更するだけで」柔軟に対応できるようになります．

例えば，本誌2016年8月号で紹介したマイコンとしても使えるWi-Fi付きSDカードFlashAirでは，Luaスクリプトの実行が可能で，さまざまなアプリケーションに仕立てることが可能です[(2)]．

ソフトウェア構成

MicroPythonを使いたい場合，ターゲット・マイコン用にCやC++で実装（移植）したMicroPythonをコンパイルしてターゲット・マイコン上で動作させることになります．

図1にソフトウェアの構成を示します．プリミティブな機能をCやC++で実現し，それらを組み合わせたサービスをMicroPythonのオブジェクトにバインディングして，呼び出す形をとります．

システムの主な機能はCやC++で実装し，これをMicroPythonから呼び出す形をとります．

MicroPython側から見るとMicroPythonスクリプトが主ですが，システム全体で見ると一般的な組み込みシステムと同様，CやC++を中心にシステムが構成

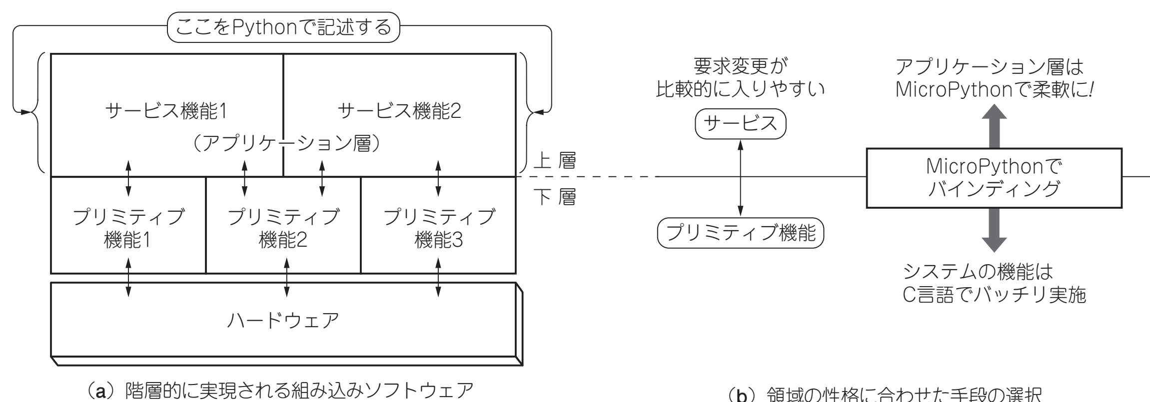


図1　マイコン用PythonスクリプトMicroPythonを使うときのソフトウェア構成
システムのコアとなるプリミティブ機能はC言語で記述し，アプリケーション層に近い部分をMicroPythonで記述することで，変更要件に応じて柔軟に対応できる

されています．これは，組み込みシステムの設計者から見ると大変好都合で，例えばハードウェアに依存するような機能を追加する場合，単に従来通りの開発をCやC++で行い，後からMicroPythonのオブジェクトにバインディングすればよいことになります．

使用可能なライブラリ

ライブラリはMicroPythonに組み込むために作られており，コアとなる機能を提供すると共に，標準Pythonライブラリの代わりに使用できるように意図されています．

表1にimport文で使用可能になるライブラリを，**表2**に固有機能を提供するライブラリを示します．モジュール名にはuで始まる名前が付けられていますが，uを付けなくてもしても使えるようになっています．

```
例：ujson <-> json
```

まずはPCの仮想マシン上でMicroPythonを動かす

● 準備

それでは，まずは実際に感じをつかむ意味でMicroPythonをPC上で実行してみましょう．手元にPCさえあれば，特別に何かハードウェアを準備する必要もありません．

ビルド環境にはUbuntu 16.04を用いました．

▶ステップ1：インストール

初めにビルドに必要となるパッケージをインストールします．

```
sudo apt-get install build-essential libffi-dev pkg-config
```

▶ステップ2：ソースなどを持ってくる

次にgitを使ってMicroPythonのプロジェクトをcloneします．

```
git clone https://github.com/micropython/micropython.git
```

▶ステップ3：サブモジュールをmakeする

cloneしたプロジェクトのディレクトリに移動します．unixディレクトリはUNIX環境で実行可能なバイナリを生成するために必要なリソースが収められているディレクトリです．

```
cd micropython/unix
```

次にaxtlsターゲットを指定してmakeを実行します．このaxtlsターゲットは，MicroPythonに必要となるサブモジュールをビルドするためのものです．

```
make axtls
```

先の例ではclone時に再帰的にcloneする指定をしていません．この場合，以下のようなメッセージが表示され，自動的にサブモジュールをフェッチしてくれます．

```
You cloned without --recursive, fetching submodules for you.
(cd ..; git submodule update --init --recursive)
```

▶ステップ4：MicroPython本体をmakeする

次にMicroPython本体をビルドします．

```
make
```

ビルド・ログを後で見返したければ，以下のように標準出力とエラー出力をまとめた上でteeを経由してファイルに保存できます．

```
make 2>&1 | tee BUILDLOG-UNIX.TXT
```

これで実行準備が整いました．

● 対話型インタプリタで実行してみる

Pythonには，対話型インタプリタ[注1]が付属しています．MicroPythonも例に漏れず対話型インタプリタでスクリプトを実行できます．

コマンド・ラインからMicroPythonの対話型イン

注1：対話型インタプリタ・バージョンv1.8.1-83-g2f7ebf1 on 2016-06-25.

表1　import文で使用可能になるMicroPythonライブラリ

モジュール名	提供される機能
cmath	複素数計算
gc	ガベージ・コレクション
math	計算
select	ストリームのイベント待ち
sys	システム固有機能
ubinascii	バイナリ/ASCII変換
ucollections	collection/container
uhashlib	ハッシュ・アルゴリズム
uheapq	ヒープ・キュー・アルゴリズム
uio	入出力ストリーム
ujson	JSONエンコーディングおよびデコーディング
uos	基本的なOSのサービス
ure	正規表現
usocket	ソケット
ustruct	プリミティブ・データ型のパックとアンパック
utime	時間関連機能
uzlib	zlibデータ圧縮機能

表2　固有機能を提供するライブラリ

モジュール名	提供される機能
machine	ボード関連機能
micropython	MicroPython内部のアクセスとコントロール
network	ネットワークの設定
uctypes	バイナリ・データ・アクセス

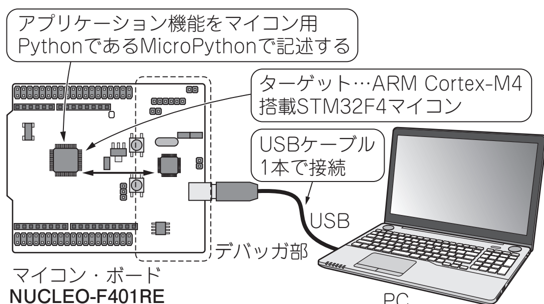


図2 32ビットARMマイコン上でMicroPythonを使うときの接続図
USBケーブル1本で電源供給，書き込み，シリアル・コンソールができて超お手軽!

タプリタを実行します．

```
./micropython
```

実行すると，プロンプトが表示されますので，変数に値を入力してprint文を呼び出してみます．

```
>>> a=1
>>> b=3
>>> c=5
>>> d=7
>>> print(a+b+c+d)
16
>>>
```

32ビットARMマイコン上でMicroPythonを動かす

次に32ビットARMマイコン上でMicroPythonを使ってみます(**図2**)．今回は手軽に入手可能なSTM32マイコン・ボードNUCLEO-F401REを選択しました(**写真1**)．マイコンは，STM32F401RET6が搭載されています(**表3**)．

表3 実験ボード搭載ARMマイコンSTM32F401RET6のスペック

項 目	仕 様
CPUコア	Cortex-M4F
内蔵フラッシュ・メモリ	512 Kバイト
内蔵SRAM	96 Kバイト
最高動作周波数	84MHz
処理性能	105 DMIPS，1.25 DMIPS/MHz (Dhrystone 2.1)
周辺機能	12ビットA-Dコンバータ，RTC，16ビット・タイマほか

● ソフトウェアの準備

▶準備1：コンパイルする

MicroPythonのリポジトリを展開したディレクトリにstmhalというディレクトリがあります．

このディレクトリに移動して以下のコマンドを実行します．

```
make BOARD=NUCLEO_F401RE
```

選択可能なボードを見つけるには，このディレクトリの下にあるboardsディレクトリを見てみます．

```
tree  -d boards/
boards/
├── CERB40
├── ESPRUINO_PICO
├── HYDRABUS
├── LIMIFROG
├── NETDUINO_PLUS_2
├── NUCLEO_F401RE  ←今回使うボード
├── NUCLEO_F411RE
・・・
└── STM32L476DISC
```

▶準備2-1：USBを接続して書き込みを実行する(Windowsの場合)

STM32マイコンの書き込み/デバッガ用USBドライバ「ST-Link，ST-Link/V2，ST-Link/V2-1 USB driver signed for XP，Windows7，Windows8」をSTマイクロエレクトロニクスのDevelopment Tool SoftwareのSTSW-LINK009からダウンロードしてインストールします．

次にSTM32 ST-LINK Utilityを用いて出来上がったfirmware.hexを書き込みます．

▶準備2-2：USBを接続して書き込みを実行する(Linuxの場合)

NUCLEO-F401REにはST-LINKと呼ばれるビルトインされたプログラマが搭載されており，https://github.com/texane/stlinkで公開されている書き込みツールst-flashを使って書き込みを実行できます．

```
$ sudo make BOARD=NUCLEO_F401RE
deploy-stlink
```

ボードの自動検出に失敗する場合，以下のようにlsusbでデバイスの所在(バス番号，アドレス)を確認した後で環境変数を設定して再実行します．

```
$ lsusb
   ・・・
Bus 002 Device 011: ID 0483:374b
STMicroelectronics ST-LINK/V2.1
(Nucleo-F103RB)
   ・・・
$ export STLINK_DEVICE="002:0011"
$ sudo make BOARD=NUCLEO_F401RE
deploy-stlink
```

次にシリアル・コンソールでターゲットに接続します．screenやminicomなど好みの端末エミュレータ

を使用します．

```
sudo minicom -b 115200 -D /dev/
ttyACM0
sudo screen /dev/ttyACM0 115200
```

▶準備3：USB仮想シリアル・ポートで接続する

ボードはUSB仮想シリアル・ポートで接続できるようになっています．ボーレートは115200に設定します．

▶準備4：接続確認

接続したらボード上のリセット・スイッチを押してみます．以下のようなプロンプトが出力されていれば動作しています．

```
MicroPython 410a502 on 2016-06-21;
NUCLEO-F401RE with STM32F401xE
Type "help()" for more information.
>>>
```

ここで，help()を入力してEnterキーを押します．操作可能な内容が端末に表示されます．

● MicroPython動作：I/O操作が可能なことを確認する

▶その1：出力確認

試しにボード上のLEDの点灯状態を変更してみましょう．以下のコマンドを打ち込むとボード上にあるLED(LD2)の点灯状態が点灯，消灯と交互に変化します．

```
>>> pyb.LED(1).toggle()
>>> pyb.LED(1).toggle()
>>> pyb.LED(1).toggle()
```

このコンソールでは上矢印キーによる履歴からの入力が可能ですので，一度入力したら上矢印キーとEnterを入力するだけで実行できます．

▶その2：入力確認

入力(GPI)の状態を読み取ってみます．

```
>>> g = pyb.Pin('PC13', pyb.Pin.IN)
>>> print(g.value())
1
>>> print(g.value())
0
```

マイコン用PythonスクリプトMicroPythonが使える状態になったことが，最低限確認できたかと思います．

MicroPythonの良さ1…プログラムの変更が簡単

● OSのモジュールをインポートする

import osとしてOSのモジュールをインポートすると，ファイル・システムを扱うことができます．

試しにNUCLEO-F401REを使ってOSの動作を見てみます．起動してからおもむろにimport osをタイプし，os.listdir()と入力してEnterキーを押します．すると，フラッシュ・メモリ上に構成されたファイル・システムのファイルをリスト表示できます．

```
>>> import os
>>> os.listdir('/')
 ['flash']
>>> os.listdir('/flash')
 ['main.py', 'pybcdc.inf', 'README
.txt', 'boot.py']
```

既存のファイルを開いて中身を表示してみます．

```
>>> f = open('/flash/main.py', 'r')
>>> print(f.read())
# main.py -- put your code here!
```

● ブート後の実行内容を変更する

ここで，試しにブート後に実行する内容を変更してみます．

```
>>> f = open('/flash/boot.py', 'w')
>>> f.write('import machine¥n')
15
>>> f.write('import pyb¥n')
11
>>> f.write('pyb.main(¥'main.
py¥')¥n')
20
>>> f.close()
```

f.write時に出力される数字は書き込まれたバイト数です．正しく書き込まれたかどうか確認してみます．

```
>>> f = open('/flash/boot.py', 'r')
>>> print(f.read())
import machine
import pyb
pyb.main('main.py')
```

boot.pyからmain.pyを呼び出すコードを書いたので，次にmain.pyを作ります．

```
>>> f = open('/flash/main.py', 'w')
>>> f.write('print(¥'CQ Inter
face.¥')')
32
>>> f.close()
```

これも内容を確認します．

```
>>> f = open('/flash/main.py', 'r')
>>> print(f.read())
print('CQ Interface.')
>>> f.close()
```

ボード上のリセット・スイッチを押して端末の出力を観察してみましょう．以下のように表示されたら，

boot.pyとmain.pyの書き換えに成功しています.

```
CQ Interface.
MicroPython v1.8.1-83-g2f7ebf1 on
2016-06-26; NUCLEO-F401RE with
STM32F401xE
Type "help()" for more information.
>>>
```

このように，コールド・スタート時の挙動をMicroPythonスクリプトを書き換えただけで変更できました．これは，MicroPythonの下層でファイル・システム・ドライバが動作し，かつそれをMicroPythonから呼び出すことのできる層があるおかげです．MicroPythonのスクリプトを書くユーザは，それを意識せずとも簡単に扱えます.

MicroPythonの良さ2…機能を簡単に拡張できる

MicroPythonでは，特定の動作をする機能をオブジェクトを単位として簡単に拡張できます．実際に既存の実装に新しいオブジェクト・タイプを追加して機能を追加してみましょう．MicroPythonが動作しているマイコン・ボードに接続して，コンソール上で何も入力しない状態でTabキーを押します.

```
>>>
machine             __name__            pyb
```

machine，__name__，pybと三つのモジュールが表示されます．今回はpybにCQという新しいオブジェクトを追加し，付随して新たに追加するC言語の関数に橋渡して操作できるようにしてみます.

ちなみに，ある領域から別の領域に何らかのメカニズムを介して機能や情報の接続を行うことを，プログラミングの世界でバインディングと呼んでいます．今回の場合，MicroPythonのオブジェクトに対する操作をC言語で実装された関数への実行へバインディングすることになります．MicroPythonからCの世界へのバインディングがどのように行われるのかを見てみます.

● 事前準備

▶準備1：空のファイルを追加

初めにstmhalディレクトリの下にcq.cとcq.hを空のファイルで作ります．次にstmhalディレクトリの下にあるMakefileのSRC_Cのリストにcq.cを加えます.

```
SRC_C = ¥
      ・・・
      servo.c ¥
      dac.c ¥
      adc.c ¥
      cq.c ¥   ←追加した
      $(wildcard boards/$(BOARD)
/*.c)
```

次にstmhalディレクトリの下にあるmodpyb.cにinclude "cq.h"を追加して事前準備が完了です.

```
#include "accel.h"
  ・・・
#include "usb.h"
#include "cq.h" ←追加した
#include "portmodules.h"
```

▶準備2：最低限のモジュール用コードを記述する

初めに，cq.hにオブジェクトを参照するためのコードを追加します．これを見て分かるようにpyb_cq_typeはmp_obj_type_tです.

```
extern const mp_obj_type_t pyb_cq_
type;
```

次にcq.cに最低限の記述を追加してみましょう.

```
#include "py/nlr.h"
#include "py/runtime.h"
#include "py/mphal.h"
#include "cq.h"
const mp_obj_type_t pyb_cq_type = {
      { &mp_type_type },
      .name = MP_QSTR_CQ,
};
```

mp_obj_type_tのフィールドに対してさまざまな属性を設定することで，MicroPython側がそれに応じた処理を行ってくれるように設計されています.

上記は，CQという名前の型オブジェクトが宣言されたことを示しています.

ここまでの作業の流れは，以下のようになっています.

(1) 空のcq.cとcq.hを作る.
(2) コンパイルするためにMakefileにcq.cを追加する.
(3) 最低限の実装を追加する.

という状態です.

実際にこの状態で既にコンパイルが可能になっていますので，make BOARD=NUCLEO_F401REとしてコンパイルが通ることを確認します．ここまでで最低限の下地が整ったことになるのですが，実際にモジュールを呼び出すための実装はこれからです．続く節では，pybに追加する形で呼び出しができるようにします.

先ほど編集したmodpyb.cには，pyb_module_globals_tableという変数があります．これはmp_map_elem_t構造体のリストで，QSTRとオブジェクト・タイプを関連付けるためのテーブルになっています.

このテーブルに以下の記述を追加します.

これはMP_QSTR_CQがpyb_cq_typeに関連付

けられることを意味します.

```
STATIC const mp_map_elem_t pyb_
module_globals_table[] = {
  . . .
  #if defined(MICROPY_HW_LED1)
   { MP_OBJ_NEW_QSTR(MP_QSTR_LED),
(mp_obj_t)&pyb_led_type },
  #endif
   { MP_OBJ_NEW_QSTR(MP_QSTR_CQ),
(mp_obj_t)&pyb_cq_type },  ←追加した
   { MP_OBJ_NEW_QSTR(MP_QSTR_I2C),
(mp_obj_t)&pyb_i2c_type },
   { MP_OBJ_NEW_QSTR(MP_QSTR_SPI),
(mp_obj_t)&pyb_spi_type },
   { MP_OBJ_NEW_QSTR(MP_QSTR_UART),
(mp_obj_t)&pyb_uart_type },
  . . .
```

pyb_cq_typeはcq.hで以下の宣言をしており, 実体はcq.cに実装されています.

```
extern const mp_obj_type_t pyb_cq_
type;
```

● 新しい機能を実装する

先ほどのpyb_cq_typeの実体に対して, 新しい機能を追加していきます.

(1) .printフィールドは表示処理が必要になったときにMicroPython側から呼び出される関数です.
(2) .make_newはインスタンス生成処理が必要になったときにMicroPython側から呼び出される関数です.
(3) .locals_dictはCQオブジェクト内でオブジェクトを参照するときに呼び出される関数です.

このように, 特定のオブジェクトに対する処理は, mp_obj_type_tに定義されている関数ポインタに所望の関数ポインタを設定しておくことで処理されるようになっています.

```
const mp_obj_type_t pyb_cq_type = {
      { &mp_type_type },
      .name = MP_QSTR_CQ,
      .print = cq_obj_print,
      .make_new = cq_obj_make_new,
      .locals_dict =
          (mp_obj_t)&cq_locals_dict,
};
```

手始めに.printに設定した関数の実装例を見てみましょう. この関数には, mp_printfで使用するための構造体mp_print_t, オブジェクト構造体へのポインタmp_obj_t, 表示処理内容を示すmp_print_kind_tが渡されます. mp_obj_t self_inは, MicroPython側でpyb_cq_obj_tへのポインタであることが保証されていますので, そのままキャストできます.

```
void cq_obj_print(const mp_print_t
*print, mp_obj_t self_in,
            mp_print_kind_t kind) {
      pyb_cq_obj_t *self = self_in;
      mp_printf(print, "CQ(%lu)",
                         self->cq_id);
}
```

pyb_cq_obj_tにはcq_idというフィールドを加え, インスタンスの番号を格納できるようにしておきました.

```
typedef struct _pyb_cq_obj_t {
      mp_obj_base_t base;
      mp_uint_t cq_id; ←インスタンス生
成時に与えるインスタンス番号を格納できるように
した
} pyb_cq_obj_t;
```

コールバック関数がどのように用いられるのかイメージがつかめたところで, インスタンス生成を実行する.make_newを見てみましょう. この関数にはMicroPython上で与えたパラメータが渡されるようになっています.

今回はCQ(1)のようにインスタンスの番号を1から4で選択したオブジェクトを生成できるようにしました. 仮に不正なインスタンス番号が与えられた場合には, エラー・メッセージを表示します.

正常にインスタンスを返すことができる場合, 先ほどの説明から出てくるpyb_cq_objのアドレスを返す仕組みで, このpyb_cq_objはpyb_cq_obj_tです.

.printと.make_newを対に見てみると理解が進むと思います. インスタンスはpyb_cq_obj_tへのポインタ, .printに渡されるself_inもpyb_cq_obj_tでした. .make_newで返したポインタがそのまま.printでも使われるのです. これでつながりました.

```
STATIC const pyb_cq_obj_t pyb_cq_
obj[] = {
      {{&pyb_cq_type}, 1},
      {{&pyb_cq_type}, 2},
      {{&pyb_cq_type}, 3},
      {{&pyb_cq_type}, 4},
};
STATIC mp_obj_t cq_obj_make_new
(const mp_obj_type_t *type,
        mp_uint_t n_args, mp_uint_t
       n_kw, const mp_obj_t *args) {
```

```
        // check arguments
        mp_arg_check_num(n_args,
                     n_kw, 1, 1, false);
        // get CQ number
        mp_int_t cq_id =
              mp_obj_get_int(args[0]);
        // check CQ number
if (!  (1 <= cq_id && cq_id <= 4))
{
      nlr_raise(
        mp_obj_new_exception_msg_varg(
          &mp_type_ValueError, "CQ(%d)
              does not exist", cq_id));
        }
        // return static CQ object
        return (mp_obj_t)&pyb_cq_obj
                                [cq_id - 1];
}
```

● オブジェクトを参照する時に呼び出される関数

例えばCQオブジェクトに二つの操作on()とoff()がある場合について考えてみます．.locals_dictを見てみると，最終的にpyb_cq_typeは，以下のような実装になります．

```
const mp_obj_type_t pyb_cq_type = {
        { &mp_type_type },
        .name = MP_QSTR_CQ,
        .print = cq_obj_print,
        .make_new = cq_obj_make_new,
        .locals_dict = (mp_obj_t)
         &cq_locals_dict, ←新たに追加した
};
```

この.locals_dictは，CQオブジェクトが内包するオブジェクトへの参照が必要になった時に呼び出される関数です．MicroPythonのコンソールでCQ.と入力した後にTabキーを押したときに参照される辞書も同じです．初めにcq_locals_dictと実際のテーブルを関連付ける記述を追加します．

```
STATIC MP_DEFINE_CONST_DICT(cq_
locals_dict, cq_locals_dict_table);
```

● テーブルの実装

次にcq_locals_dictと関連付けられる実際のテーブルを実装します．

```
STATIC const mp_map_elem_t cq_
locals_dict_table[] = {
    { MP_OBJ_NEW_QSTR(MP_QSTR_on),
(mp_obj_t)&cq_obj_on_obj },
    { MP_OBJ_NEW_QSTR(MP_QSTR_off),
(mp_obj_t)&cq_obj_off_obj },
};
```

.locals_dictにテーブルへのポインタを直接記述するわけではないことに注意しましょう．テーブルに記述した内容は，そのままMicroPythonの機能になります．

例えば，MP_OBJ_NEW_QSTR(MP_QSTR_on)は，onというQSTRが定義されます．ユーザからonが入力として与えられた場合，cq_obj_on_objが呼び出される仕組みです．これも実際には関数とのバインディングを行うマクロで実際の関数との関連付けを行います．

```
STATIC MP_DEFINE_CONST_FUN_
OBJ_1(cq_obj_on_obj, cq_obj_on);
STATIC MP_DEFINE_CONST_FUN_
OBJ_1(cq_obj_off_obj, cq_obj_off);
```

● 関数の実装

実際に呼び出される関数の実装例を見てみます．

```
mp_obj_t cq_obj_on(mp_obj_t
                          self_in) {
      pyb_cq_obj_t *self = self_in;
      cq_state(self->cq_id, 1);
      return mp_const_none;
}
mp_obj_t cq_obj_off(mp_obj_t
                          self_in) {
      pyb_cq_obj_t *self = self_in;
      cq_state(self->cq_id, 0);
      return mp_const_none;
}
```

関数cq_stateはインスタンス番号の状態を受け取って何かを実行する関数です．まさにこの箇所がMicroPythonの世界からCの世界に橋渡しされる部分です．

◆参考文献◆

(1) MicroPythonのウェブ・サイト．https://micropython.org/
(2) 特集 切手サイズIoT無線センサ入門，Interface，2016年8月号，CQ出版社．
(3) http://wiki.micropython.org/Creating-Pyb-Modules
(4) Bill Lubanovic 著，斎藤 康毅 監訳，長尾 高弘 訳；入門Python 3，2015年12月．

なかむら・しんいちろう

Appendix 4 巨大ファイルの分割ダウンロード/アップロードをスクリプトでさせてみた

アマゾン・クラウドもPython! AWS用ライブラリboto

牧田 達郎

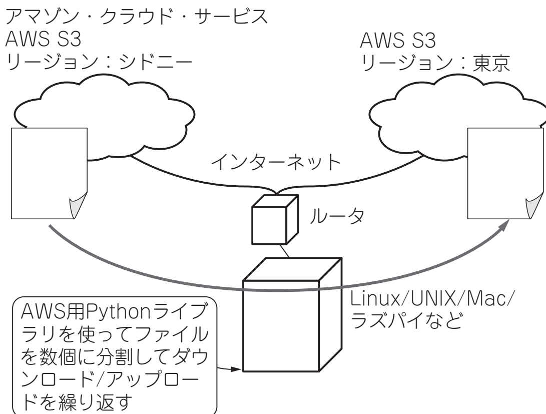


図1　実験の全体構成…クラウド・ストレージ上にあるファイルをストリーミング転送する



図2　AWSのアカウント作成サイト
Eメール・アドレスまたは携帯番号を入力することでアカウントがすぐに作成できる

クラウド・ストレージ上にあるファイルを別のクラウド・ストレージに転送する場合，一度ローカルのディスクにダウンロードしてからアップロードする手順を踏みます．ダウンロードしようと思ってもディスク・サイズに限界があって，ダウンロードできないこともあります．解決する方法としては，ファイルを分割してダウンロード/アップロードを自動で行います．小さなディスクやメモリでも数テラ・バイトの巨大なファイルを自動で転送することができます．

環境構築

● 全体の構成

図1に示すように，Amazon Web Service（以降，AWSと記載）のSimple Storage Service（以降，S3と記載）を利用して，クラウド・ストレージ上にあるファイルをストリーミング転送します．

● 必要ならアマゾン・クラウドAWSのアカウントを作成する

S3上にファイルを配置してあるファイルにアクセスするために，AWSのアカウントが必要になります．AWS上にアカウントがない場合は，下記のサイトから作成することが可能です（**図2**）．

```
https://portal.aws.amazon.com/gp/aws/developer/registration/index.html
```

2016年8月現在では，アカウントを無料で作成でき，5Gバイトのクラウド・ストレージを無料で使うことができます．

● 転送情報を取得しておく

下記の情報を事前に取得しておいてください．

- 転送するファイルを格納しているリージョンのエンドポイント
- 転送するファイルを格納しているバケット名
- 転送するファイル名
- AWSのアクセス・キー
- AWSのシークレット・キー
- 宛て先のリージョンのエンドポイント
- 宛て先のバケット名
- 宛て先のファイル名

なお，リージョンのエンドポイントは，下記のURLから入手可能です．

```
http://docs.aws.amazon.com/general/latest/gr/rande.html
```

● AWS用Pythonライブラリをインストールする

botoというAWSのAPIを発行してくれるPythonライブラリAWS SDK for Python（boto 3）をインストールします．

リスト1　アマゾン・クラウドAWS上の巨大ファイルを分割してダウンロード/アップロードするPythonプログラム
ファイル・サイズを確認して，そのファイルを指定したサイズで分割してダウンロード&アップロードする

```
#必要なライブラリをインポート・・・(1)
import math
from   boto.s3.connection import S3Connection
from   cStringIO          import StringIO

#処理に必要なパラメータを定義・・・(2)
source_host='s3-website-us-east-1'
source_bucketname='bucket-001'
source_key='obj_001'
aws_key='00000000000000000000'
aws_secret='aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa'
destination_host='s3-website-us-east-1'
destination_bucketname='bucket-002'
destination_key='obj_002'

#コネクションを生成・・・(3)
source_conn    = S3Connection(aws_key, aws_secret,
                    is_secure=False, host=source_host)

#バケット情報を取得・・・(4)
source_bucket = source_conn.get_bucket(source_
                                        bucketname)

#ファイル・サイズを取得・・・(5)
source_size   = source_bucket.get_key(source_key).size

#分割サイズを100Kバイト程度に指定し分割したバイト数を計算・・・(6)
#マシン上で使用可能なメモリのサイズを指定．5Mバイト以上の指定が可能．
chunk_size    = 5 * 1024 * 1024
chunk_amount = int(math.ceil(source_size /
                                 float(chunk_size)))

#宛て先にアップロードする際に必要となるマルチパートIDを取得・・・(7)
destination_conn    = S3Connection(aws_key, aws_
       secret, is_secure=False, host=destination_host)
destination_bucket = destination_conn.get_bucket
                             (destination_bucketname)
headers={}
headers.update({'Content-Type': 'application/octet-
                                              stream'})
multipart_object = destination_bucket.initiate_
    multipart_upload(destination_key, headers=headers)
multipart_id = multipart_object.id

#指定サイズのファイルをダウンロード&アップロードを分割した数ぶん繰り返す
                                                ・・・(8)
for i in range(chunk_amount):
    part_num = i + 1
    offset = chunk_size * i
    bytes    = min(chunk_size, source_size - offset)
    source_body = source_conn.make_request("GET",
                        bucket=source_bucketname,
                        key=source_key,
                        headers={'Range':"bytes=%d-
                              %d" % (offset, bytes)})
    fp = StringIO(source_body.read())
    destination_conn = S3Connection(aws_key, aws_
       secret, is_secure=False, host=destination_host)
    destination_bucket = destination_conn.get_bucket
                             (destination_bucketname)
    for mp in destination_bucket.get_all_multipart_
                                           uploads():
        if mp.id == multipart_id:
            mp.upload_part_from_file(fp=fp, part_
                                         num=part_num)

#アップロードが完了したら最後に完了のAPIを飛ばす・・・(9)
mp.complete_upload()
```

```
wget https://github.com/boto/boto/
archive/2.39.0.tar.gz
cd boto-2.39.0
sudo python setup.py install
```

AWS通信ライブラリbotoにはS3に限らず，EC2などのAWSのクラウド・サービス全般を操作するAPIを発行するライブラリも実装されています．ここではS3の部分を使います．

巨大ファイルの分割転送プログラム

ファイル・サイズを確認して，そのファイルを指定したサイズで分割してダウンロード&アップロードします．

プログラム・リストを**リスト1**に示します．プログラムの流れは下記のようになります．

(1) 必要なライブラリをインポートする
(2) 処理に必要なパラメータを定義する
(3) コネクションを生成する
(4) バケット情報を取得する
(5) ファイル・サイズを取得する
(6) 分割サイズを100Kバイト程度に指定し，分割したバイト数を計算する．マシン上で使用可能なメモリのサイズを指定する(5Mバイト以上の指定が可能)
(7) 宛て先にアップロードする際に必要となるマルチパートIDを取得する
(8) 指定サイズのファイルをダウンロード&アップロードを分割した数ぶん繰り返す
(9) アップロードが完了したら最後に完了のAPIを飛ばす

バケット情報の取得の際に下記のようなエラーが出た場合は，サーバの名前解決に失敗している可能性があります．転送元のリージョンのエンドポイントをもう一度確認し直してみましょう．

```
socket.gaierror: [Errno -3] Tempo
rary failure in name resolution
```

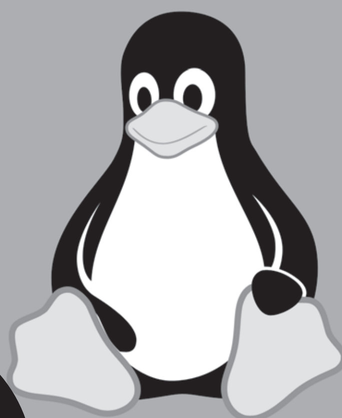
以上で，アップロードが自動で行われるようになります．

まきた・たつろう

ダウンロード・データあります

夏の2大定番入門講座

UNIX



全コンピュータ対応





第2特集

オープンソース組み合わせ時代の重要言語

IoTから! シェル再入門

執筆：中村 和敬

オフ会 「Python＆シェル探検隊（仮）」

日時：2016年10月18日（火）19～21時
場所：東京巣鴨（CQ出版社） 会費：1,000円
詳細は本誌ウェブ・サイト（http://interface.cqpub.co.jp/）

第2特集 Appendix 1

スパコンから手のひらボードまで！
コンピュータの超定番で組み合わせ自在！

オープンソース時代にますます重要！伝統的シェルのメリット

中村 和敬

なぜシェルを学ぶべきなのでしょうか．シェル自体に由来するメリットと，UNIXに由来するメリットのそれぞれの観点から挙げてみました．

シェル自体に由来するメリット

● メリット1：UNIXのパワーを全て引き出せる

シェルは最初にUNIXのユーザ・インターフェースとして開発され，現在も全てのUNIX系OSにシェルが搭載されています．シェルは本格的なプログラム言語であり，UNIXはプログラムの起動やファイルの操作といった抽象度の高い処理から，デバイスの直接的な操作のような抽象度の低い処理まで，シェルを通じて行うことができるように作られています．シェルの使い方を知ることで，スーパコンピュータから組み込み用のボード・コンピュータまで，UNIX系OSの動作するさまざまなコンピュータでシステムを作れます．

● メリット2：習得しやすく理解しやすい

シェル・プログラムの特徴は，シェル自体で処理を行うのではなく，コマンドに処理をさせるという点です．一つのコマンドだけでは目的の処理が記述できない場合は，UNIXのパイプという機能を利用して，プログラムを連係動作させるように記述します．シェルで良いプログラムを書くためのコツは，データの流れをプログラムするという考え方です．そういった考え方に基づきプログラムを行うとパイプを多用するようになりますが，そういったプログラムは開発しやすく，理解しやすいことが利点です．

● メリット3：他のプログラム言語の力も借りやすい

シェルはコマンドを呼び出し，パイプで連携させて処理を行うためのものです．そのため，Python，Rubyなどの他のプログラム言語で作成されたライブラリも，UNIXのコマンドという形に加工することで，組み合わせて利用しやすいという利点があります．シェルを利用すると他のプログラム言語の力を借りやすいのです．

● メリット4：マルチコアでの並列処理が容易

シェルのプログラムの際の考え方を守ると，初めから処理性能の高いプログラムを作成できます．スクリプト言語は一般的に処理性能が問題になることが多々あります．確かにシェルもそれ自体の処理性能は高いとはいえませんが，シェル自体は実際には処理を行いません．実際に処理を行うのは，C言語などで作成された高速なコマンドなのです．

さらに，シェル・プログラムは近年主流のマルチコア・プロセッサの性能を引き出せるという利点があります．パイプでコマンドを結合するスタイルで記述された処理は，初めからそれぞれのコマンドが並列して動作します．そのため，マルチコア・プロセッサの性能を容易に引き出すことが可能であり，シェルで記述されたプログラムは初めから高速に動作するのです．

● メリット5：分散処理の実装も容易

ネットワークで結合された多くのコンピュータを利用する分散処理も簡単に実行することができます．シェルには初めからバックグラウンド実行や，waitといったプロセス管理の機能が実装されているので，それらを使用することで簡単に並列実行が可能です．さらに，sshなどのコマンドを使用してネットワークを通じて別のコンピュータに処理をさせることも簡単にできるので，分散処理も簡単に実装することができます．

他のプログラム言語では並列処理や，分散処理のためのライブラリの使い方を学ばなければなりませんが，シェルではその必要は全くありません．

UNIXに由来するメリット

● メリット6：長い歴史を持っており安定している

UNIXは，1968年にAT＆Tのベル研究所のケン・トンプソン，デニス・リッチー，ブライアン・カーニ

ハンらによって開発されたOSです．

UNIX以前には，1964年から開発が始まったMulticsというOSがありました．Multicsは先進的なアイデアがこれでもかと盛り込まれた，まさに夢のOSでした．ケン・トンプソンらもその開発に参画していましたが，Multicsの機能の過剰さに嫌気が差し，Multicsの機能の中から本当に必要なもののみを厳選してUNIXを開発しました．

こうして開発されたUNIXは理解しやすく，それでいて利用しやすいものになりました．それ以来，UNIXは学術界から産業界まで幅広い支持を得て，現在に至るまで利用者層を拡大してきました．

現代のUNIX系OSは初期のUNIXとは異なりますが，ほとんどが20年以上継続して開発，バージョンアップを続け成熟しており，安定して動作するOSとなっています．

● メリット7：システムを作りやすい

UNIXは一貫した哲学に基づいて設計されています．それは要約するならば以下のようなものです．

- プログラムは一つのことだけを行うように作る．
- プログラムは一つの入力を読み込み，処理をして，出力するという形式で書く．
- プログラムをパイプにより協調して動作させ，大きな処理を行う．

これを補強するように，UNIXはOSを通して扱える全てのものを，仮想的なファイルとして扱えるようにしてくれます．これらの工夫により，UNIX上ではデータの流れという観点からシステムを捉えて，開発するということが可能になりました．データの処理からデバイスの制御まで，一貫した同じ考え方で取り組めるようになったのです．UNIXをベースに開発を行うと，開発するために必要な知識の量を抑え，また理解しやすいシステムを作れます．

● メリット8：オープンソースが多い

UNIXはその初期からソースコードが公開されていました．当時，AT＆T社は1956年の独占禁止法の和解判決合意により，コンピュータ産業への進出を禁じられていたためです．そのため，希望者は誰でもUNIXのソースコードを入手することができました．これにより，UNIXは，米国の多くの大学で教育，研究用のOSとして使われるようになりました．バグの修正や，新機能の追加をしたソースコードが，ベル研究所とソースコードの提供を受けた人々との間で活発にやり取りされ，一つのコミュニティを築き上げました．

その後オリジナルのUNIXのソースコードが非公開になるなど，さまざまな紆余(うよ)曲折がありましたが，現在では多くのUNIX系OS，そして多くのソフトウェアがオープンソース・プロジェクトとして開発されています．既存のプログラムの仕組みを知りたいときなどは，いつでも誰でもソースコードを見て内部の仕組みを知ることができます．

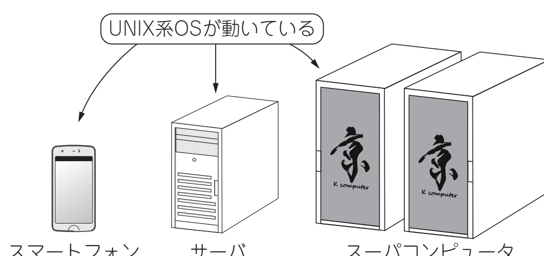


図1　UNIX系OSはさまざまなハードウェアで動作する

● メリット9：さまざまな分野で使われてきており，さまざまなハードウェアで動作する

UNIXは大学の教育研究の分野で支持を得て，さまざまな研究で利用されてきました．さらにそういった大学で学んだ学生が社会に出るにつれて，製品開発の分野にも進出していきました．近年のハードウェア性能の向上により，組み込み分野でも利用されるようになってきています．

その幅広い利用範囲のために，UNIX系OSはさまざまなハードウェアで動作します(**図1**)．例えばスーパコンピュータの「京」のOSにはLinuxが採用されています．2001年にはアップルのOS，OS Xのベースに採用されました．2008年にはスマートフォン用OSのAndroidのベースにLinuxが採用されています．

このように，UNIX系OSはスーパコンピュータやサーバ機からスマートフォン，組み込み機器まで，さまざまなハードウェアで利用されています．UNIXを使えれば，開発の幅が広がることでしょう．

● メリット10：これまで開発されたプログラムの蓄積も豊富で成熟している

UNIXはさまざまな分野，ハードウェアで長年にわたって利用されてきました．それに伴いさまざまな分野の問題を解決するプログラムが開発されてきており，しかも成熟しているという利点があります．現在でも最先端の研究の分野でも安定して支持を得ており，最新の研究の成果はUNIXで動くプログラムとして開発されることがほとんどです．UNIXを利用することで人類の英知の蓄積を利用できるのです．

なかむら・かずたか

第2特集 Appendix 2

仮想マシン不要のBash on Ubuntu on Windows出た!

ついに全コンピュータ制覇! Windows正式対応シェルの世界

中村 和敬

2016年3月30日，マイクロソフトは同社の主催する開発者向けカンファレンスBuild 2016において，Windows 10でシェルのBashが動作する仕組みであるBash on Ubuntu on Windowsを発表しました．これはその名の通り，Windows上でLinuxディストリビューションの一つであるUbuntuを動かし，Bashを通じて操作ができるようになるという仕組みです．2016年8月2日に実施されたWindows 10の大型アップデートAnniversary Updateにて，この機能がサポートされました．Bashは多くのLinuxディストリビューションで標準のシェルとなっています．

● 広く使われているUNIX環境をWindowsにも

マイクロソフトがBash on Ubuntu on Windowsを開発するに至ったのは開発者の便宜を図るためです．

UNIX系OSは，情報科学研究/教育や，インターネットの構築，ネットワーク・サービスの開発で以前から使われていました．一方で，UNIX系OSを日常的に使用するためには，日本語やGUIといった環境の整備が必要でした．そのためにかつては複雑なセットアップのステップを踏む必要がありましたが，近年ではディストリビューションの多様化やパッケージ・システムの充実などにより大幅に解決され，Linuxのシェアが徐々に伸び始めています(1)．

特に象徴的な出来事は，2001年にMacのOS XとしてBSDベースのUNIXが採用されたことでしょう．これにより，特にオープンソース・ソフトウェアやスマートフォンのアプリ開発の分野にかかわる人々の間で，Macは大きくシェアを伸ばしました．UNIX上でのシステム開発で使用していたアプリケーションがMacの上でも動作するようになり，Linux，iOS，Android向けのアプリが開発しやすいということが挙げられるでしょう．

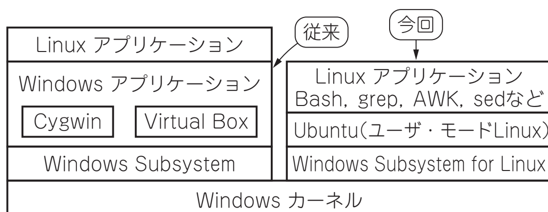


図1　Windowsでシェル&Linuxが動く仕組み

こういった状況を踏まえてWindowsも後れをとりたくないという狙いがあると思われます．あくまで開発用なので現状では幾つかの制約はあるものの，標準的なUNIXコマンドライン・ツールについては，積極的にサポートしていく方針のようです．

● WindowsでLinuxが動作するメカニズム

Bash on Ubuntu on Windowsの仕組みを**図1**に示します．

通常のWindowsアプリケーションを使用する場合，Windows Subsystemという階層を通じて，Windowsカーネルの機能を利用します．

Bash on Ubuntu on Windowsの場合，まずLinuxを動作させるために，Windows Subsystem for Linuxという階層を設けて，Linuxをユーザランド・アプリケーションとして動作させます．ここで用いるLinuxディストリビューションとしては，マイクロソフトはUbuntuを標準としてサポートしています．

これまでWindows上でLinuxを動作させるためには，Virtual Boxなどの仮想マシンを使用する方法がありました．一方でWindows Subsystem for Linuxを用いる方法は，そういった方法よりはるかに高速に動作しますし，なんの設定もなくWindowsのファイル・システムをシェルからのぞくことができます．

また，Cygwinのように疑似的にUNIX系OSの環境を再現する方法もありました．この場合，Cygwin用にプログラムをコンパイルしなければならなかったのですが，Windows Subsystem for Linuxを用いる方法であればUbuntuで動作するパッケージがそのまま動作します．

このようにWindows Subsystem for Linuxにより，これまでより格段にシェルとの親和性が上がるといえるでしょう．

なかむら・かずたか

◆参考・引用*文献◆

(1) Usage share of operating systems. https://en.wikipedia.org/wiki/Usage_share_of_operating_systems

第2特集 第1章

名刺サイズ・コンピュータにもよく使うしIoT向き

やり直しのための シェル入門

中村 和敬

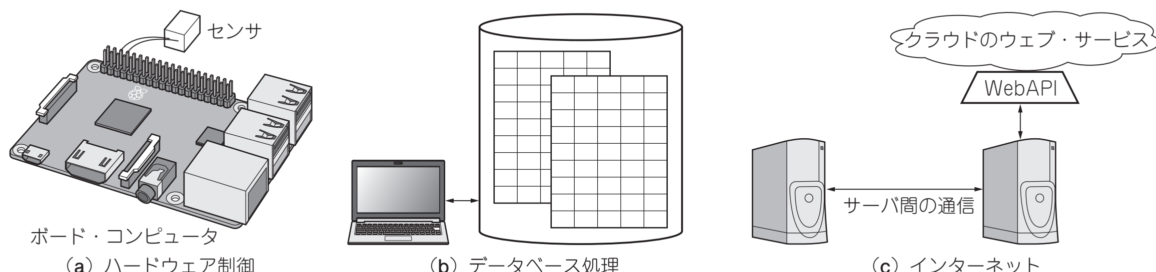


図1 伝統的なシェルができることは意外と(?) IoT時代向き!

本稿では，UNIXの中でのシェルの位置づけ，他のプログラム言語との違い，シェルでできること，シェル・スクリプトでシステムを作る場合の考え方などを紹介します．

メリット

近年高性能なCPUを搭載し，組み込み用途向けのUNIX系OSが動作するボード・コンピュータ注1が注目されています．ラズベリー・パイではLinuxのディストリビューションのDebianをアレンジしたRaspbianが動作します．

UNIXのCUI(Character-based User Interface)はシェルです．

● 全てのUNIXシステムで使える

シェルは最初にUNIXのユーザ・インターフェースとして開発され，現在も全てのUNIX系OSにシェルが搭載されています．現在のUNIX系OSではリッチなGUI(Graphical User Interface)も利用できるようになっていますが，OSの設定や，システム開発でよく使われるソフトウェアの設定などでは，シェルを通じた操作が必要です．

注1：インテルEdisonボードに搭載されているLinuxなどは，busyboxという機能が制限されたプログラムを利用しているため，本稿の内容は当てはまらない部分がある．

また，シェルは本格的なプログラム言語であり，UNIXはプログラムの起動やファイルの操作といった抽象度の高い処理から，デバイスの直接的な操作のような抽象度の低い処理までできるように作られています(図1)．シェルの使い方を知ることで，スーパコンピュータから組み込み用のボード・コンピュータまで，UNIXの動作するさまざまなコンピュータでシステムを作ることができます．

シェルで記述されたプログラムは，シェル・スクリプトと呼ばれます．

● シンプルで理解しやすい

シェル・プログラミングの特徴はUNIXのパイプという機能を多用する点です．パイプはプログラムを連係動作させるための仕組みです．

パイプを用いたシェル・プログラムはシンプルで理解しやすく，開発しやすいことが利点です．

● 他のプログラム言語で作成されたライブラリが使える

Python，Rubyなどの他のプログラム言語で作成されたライブラリも，UNIXのコマンドという形に加工することで，組み合わせて利用しやすいという利点があります．

● 並列処理を実現しやすい

近年のCPUは複数の命令実行ユニットの搭載され

表1　お勧めシェル一覧

名　称	正式名称	特　徴
ash	Almquist Shell	小型で高速なのが特徴．FreeBSDなど，BSD系のUNIXで標準のシェルとなっている
Bash	Bourne Again Shell	コマンドラインでの編集機能が充実．多くのLinuxディストリビューションで標準のシェルとなっている
Zsh	Z Shell	Bourneシェル系では最後発で，最強を目指して開発されたシェル．プログラマブルな補完機能や強化されたファイル名展開機能などの，既存のシェルをはるかに超える強力な機能が特徴

ているマルチコア・プロセッサが主流になってきています．例えば，ラズベリー・パイ2と3には，4コアのプロセッサが搭載されています．

シェルによりパイプを用いたプログラミングを行うと，マルチコア・プロセッサの性能を容易に引き出すことが可能であり，簡単に高速なプログラムを書くことができます．

おさらい：最低限の使い方

● 本特集で使うシェル…Linuxで標準Bash

シェルにはいろいろな種類があります．文法の違いで大きく分けてBourneシェル系とCシェル系(csh，tcshなど)に分けられます．また，それらに当てはまらないものも開発されています(es，scsh，fishなど)．これから学ぶのであれば，**表1**に示すBourneシェル系のどれかが良いと思います．Bourne系シェルはUNIXの標準規格であるPOSIXでも定義されており，また現在もっとも使用されている系統のシェルでもあるためです．

本稿ではBashを使います．

● プロセス一覧を表示してみる

シェルによる機能の組み合わせがどのようなものなのか，実際にシェルを通じてプログラムを呼び出しながら説明しましょう．実験環境としてはラズベリー・パイを使用します(**図2**)．

例として，プロセスの一覧を出力する場合を考えてみます．この場合，psコマンドを以下のように呼び出します．

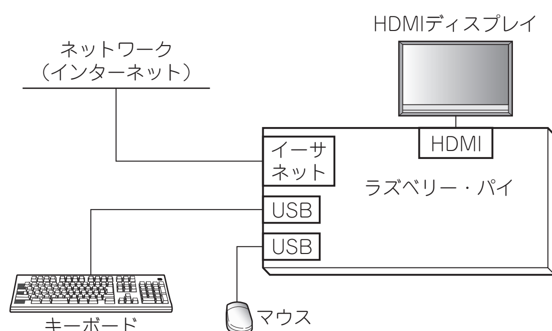


図2　シェルの機能を試すための実験環境

```
ps axl
```

● パイプを試す

出力されるプロセス一覧はたいていの場合長いので，ターミナルの1画面の中に収まりません．これ以外でも，ほとんどのUNIXコマンドは大量の出力を行う場合でも，出力を単にターミナルに吐き出すだけで，ユーザが出力を確認しやすいように配慮したりしません．なぜなら，コマンドを作成する段階ではユーザが出力の何を確認したいのかを知ることは不可能だからです．

ユーザは出力の全てに目を通したいのかもしれませんし，先頭の何行かだけ確認できればよいのかもしれません．またあるいは，特定のキーワードを含む行だけを確認したいのかもしれません．こういった機能全てを一つのコマンドの中に作り込むこともできますが，同じような機能をさまざまなコマンドの中に作り込むことになり，生産性，保守性共に悪化してしまいます．

そこでUNIXでは，それぞれのコマンドの機能は限定する一方，パイプというコマンドを組み合わせる機能を提供することで，それぞれのコマンド開発の生産性や保守性を維持しつつ，機能性を高めています(**図3**)．

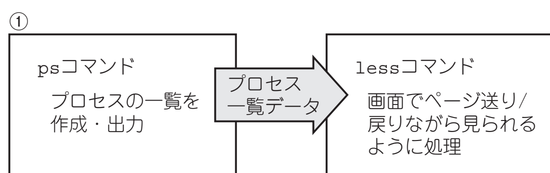
先ほどのpsコマンドの例であれば，以下のような組み合わせが定番です．

```
ps axl |less
ps axl |grep process
```

psコマンドの後に，「|」に続けて別のコマンドを記述して他のコマンドと組み合わせます．「|」がパイプを使用するための記述です．

1行目はpsコマンドの出力を，lessというコマンドに流し込むという処理です．lessはページャと呼ばれる種類のコマンドです．ページャはターミナルからあふれてしまうような大きなデータを，上下に移動しながら閲覧するためのコマンドです．kキーとjキーで上下に移動，qキーで終了させます．

2行目はgrepというコマンドに出力を流し込んでいます．grepは引き数で指定されたパターンに一致する文字列を含む行を出力するコマンドです．この例ではprocessという文字列が含まれる行を出力します．特定の名前のコマンドに関する情報を出力すると



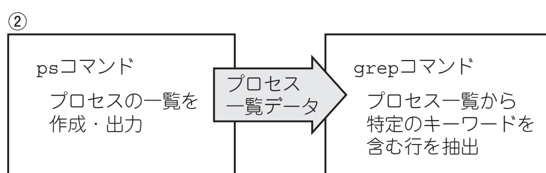


図3　コマンドを組み合わせて処理を作る

きなどに役立ちます．

● 結果を見やすく表示する

lessやgrepは独立したコマンドなので，psコマンド以外にもさまざまなコマンドと組み合わせて使用できます．例えば，2行目のgrepを使用したコマンドの出力結果がやはりたくさん出てきてしまう場合も，さらにlessをつなげて以下のように呼び出すことで全ての出力を閲覧することができます．

```
ps axl |grep process |less
```

このようにシェルを通じてコマンドをパイプで結合して，さまざまな処理を行うことができます．

パイプを利用することのもう一つの利点として，パイプで結合されたコマンドは，全てプロセスとして起動され，並列に動作することが挙げられます．そのため，シェルで記述された処理は初めから並列で処理を行います．近年のプロセッサは複数の命令実行ユニットの搭載されているマルチコアが主流になってきていますが，シェルはその処理性能を容易に引き出すことができます．

興味深い点は，シェルもコマンドであり，標準入力からデータを読み込み，処理をして，標準出力に書き出す，フィルタ・プログラムとして作成されているということです．シェルは標準入力からシェル・スクリプトを読み込み，標準出力にその結果を出力します．通常シェルへの入力はユーザからのキー入力であり，出力はユーザの見ている画面です．しかし，ファイルや他のコマンドからの出力をシェルに対する入力とし，シェルからの出力をファイルや他のコマンドへの入力とする形でシェルを利用することも可能なのです．

システム開発の考え方

UNIXとシェルは優れた仕組みですが，それはカーネルからコマンドに至るまで，独特の設計思想が貫かれているからです．その設計思想は，UNIX哲学などと呼ばれています．以下ではUNIX哲学を，UNIX上でシェルを用いてシステムを開発するという観点から見ていきましょう．

● データはテキスト・ファイルとして保管しよう

システムを開発する際には，いろいろなデータを取り扱う必要があります．これらのデータは極力テキスト・データの形，さらに可能ならば行指向のデータで表現するようにします．行指向データとは，表のような，1行に1組のデータが記述されているデータのことです．

テキスト・データであれば，特段のツールなしでもユーザが直接読んで理解することが可能です．他のシステムにデータを移行する際にも，データ形式の変換プログラムが必要ありません．データをテキスト形式で表現することで，バイナリであれば考慮しなければならないアライメントやエンディアンネスといった微妙な問題を全て回避できます．

また，UNIXのコマンドの多くは行指向のテキスト・データを処理するように作られています．システムで用いるデータを極力行指向のテキスト・データの形式で保持することで，UNIXコマンドをシステム開発に利用できるようになります．

● UNIXの提供する部品をシェルで組み合わせて作ろう

UNIXはさまざまなコマンドやデーモンによるサービス，デバイス・ファイルなどの形で，さまざまな機能を提供しています．シェルでシステムを開発する際には，パイプなどを利用してこれらの機能を組み合わせて開発します．シェルを利用するとUNIXの提供する機能を利用できるので，素早くシステムを開発することができます．

また，シェルは対話的環境なので，入力と処理と出力の関係を確認しながら，目的の処理を順を追って記述していくことができます．そして一通りの処理を記述し終わった時点でシェルに入力したコマンドの履歴を表示し，履歴を編集することでシェル・スクリプトを開発することができます．これはいわば，シェルを利用するとデバッグしながら開発ができるということであり，素早く正しいプログラムを開発することができます．

シェルを利用して開発するもう一つの利点は，さまざまなUNIX上で動作するシステムを容易に開発できることです．全てのUNIXはPOSIXという共通規格

コラム　困ったときの「man」だのみ

中村 和敬

UNIXの便利なコマンドとして，manコマンドがあります．なにか新しいコマンドが出てきたが，使い方を詳しく知りたいというときには，

`man <コマンド名>`

と入力してみましょう．するとマニュアルを参照することができます．

UNIX関連の書籍などでは，オンライン・マニュアルは“man(1)”のような形式で参照されます．これはマニュアル第一節の“man”の項目という意味で，

`“man 1 man”`

というコマンドを実行することで対応するページを読むことができます．

manコマンドはマニュアルを閲覧するためにページャと呼ばれる種類のプログラムを使用しています．

ラズベリー・パイのmanコマンドはデフォルトでlvページャを使用しており，jキーでページを上に，kキーで下に移動，qキーで終了することができます．詳しくは“man man”や“man lv”などで調べてみましょう．

に基づいて開発されています．そのためPOSIXに準拠するようにしてシェル・スクリプトを記述することで，高い移植性を確保できます．

● 部品が足りない! コマンドを作ろう

UNIXの提供する機能では，目的のシステムを構築できないか，あるいは，できてもシェル・スクリプトの実装が非常に複雑になり保守が難しくなってしまう場合があります．そういった場合に初めて新しいコマンドを開発します．コマンドの開発には，好きなプログラム言語を用いることができます．C言語はもちろん，RubyやPythonなどのスクリプト言語や，もちろんシェル・スクリプトでも構いません．

コマンドを開発する際には，幾つかのルールがあります．最初に新しいコマンドは単純な一つの機能を提供するように設計することです．たくさんの機能が必要な場合は，たくさんのコマンドを作成して，組み合わせて利用します．決して必要な機能の全てを一つのコマンドに実装してはいけません．機能を一つに絞ることで，コマンドのプログラムはシンプルなものになります．これにより開発が容易になり，素早く品質の高いコマンドを開発できるようになります．

また，コマンドは可能な限りフィルタとして作成するということが上げられます．標準入力を読み込み，標準出力に出力するという形でコマンドを作成することで，パイプにより他のコマンドを組み合わせることが可能になります．

これに関連して，対話的なインターフェースは排除することも挙げられます．対話的なインターフェースとは，例えばファイルを削除するプログラムであれば，ファイルを削除しようとするたびにユーザに確認を求めるプロンプトを出すような仕組みのことです．こういった機能はコマンドを部品として利用する際の妨げとなり，コマンドの活用を妨げます．どうしても必要な場合はオプションの形などで実装します．

機能が絞られていると汎用性が高くなるので，より多くの状況で，さまざま他のコマンドと組み合わせて利用できるようになります．ユーザもコマンドの機能や使い方を簡単に覚えることができるので，一度作ったコマンドが無駄になりません．このようなルールにのっとって開発されたコマンドはまたUNIXの一部となり，後々の開発に利用できるので，UNIXはますますパワーアップしていくのです．

このように，UNIX哲学にのっとって，システムを開発することで，個々のシステム開発はUNIX自体の機能の拡張につながり，将来別のシステムを開発する際にも無駄にならないのです．

◆参考・引用*文献◆

(1) Brian W. Kernighan，Rob Pike，石田 晴久 翻訳；UNIXプログラミング環境，アスキー，1985年.
(UNIXがどのような考えに基づいて開発されたか，どのように使用することを想定されているかについて，UNIXの開発者が書いた書籍)

(2) Mike Gancarz，芳尾 桂 訳；UNIXという考え方—その設計思想と哲学，オーム社，2001年.
(UNIX哲学がまとまっている)

(3) 上田 隆一，USP研究所 監修；シェルプログラミング実用テクニック，技術評論社，2015年.
(シェル・スクリプトの入門書)

(4) 松浦 智之，USP研究所 監修；全てのUNIXで20年動くプログラムはどう書くべきか，シーアンドアール研究所，2015年.
(シェル・スクリプトで高い可搬性，可用性を実現するための記述の仕方)

なかむら・かずたか

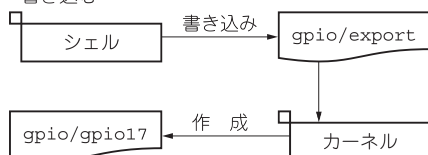
第2特集 第2章

ステップ・バイ・ステップでメカニズムを確認！

シェルからのハード操作超入門

中村 和敬

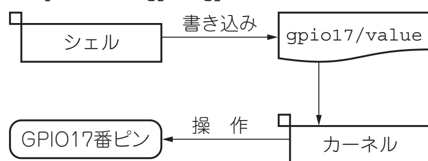
①/sys/class/gpio/exportに使用するGPIOポート番号を書き込む



②カーネルが書き込みを検知して，操作用のデバイス・ファイル群を作成

(a)ステップ1：デバイスファイルを作成する

③/sys/class/gpio/gpio17/valueに値を書き込み



④カーネルが書き込みを検知して，対応する操作を実行

(b)ステップ2：デバイス・ファイルを操作（=ハードウェアを操作）する

図1　デバイス・ファイルを通じたGPIOポート制御の仕組み

汎用OSボード・コンピュータの特徴

●マイコンと比べてソフト開発の比率が高い

Linuxが動作する代表的なボード・コンピュータとして，ラズベリー・パイがあります．GPIOやI^2Cといった組み込み向けのインターフェースから，イーサネットやUSBといったPC向けのインターフェースまでそろっていて，ちょっとしたガジェットを作る際のベースとして，定番になった感があります．

ラズベリー・パイのようなLinuxベースのボード・コンピュータを利用する場合，マイコンを用いた機器開発と比べてソフトウェア開発の比率が格段に高まります．

● ムダにたくさん書く必要はない…ハードの操作はむしろ簡単

OSが載っているためにできることが制限されたように思うことがあるかもしれません．

しかし，ラズベリー・パイの標準のOS，Raspbianは，ちょっと気の利いた小物を開発する際に便利なさまざまな機能を提供しています．シェルを通じてそれらを組み合わせることで，初心者でも簡単にさまざまな機能を持った機器を作ることができます．

また，Raspbian上からも，簡単な手順でGPIOなどの低レベル・ポートを直接操作することができます．こういった操作はシェルを通じて行うことができるので，対話的に周辺機器を操作して動作を確認しながら開発を進めることができます．

つまりシェルの使い方を学ぶことで，一からプログラムを書かなければならない場合と比べてはるかに簡単に開発を進めることができます．

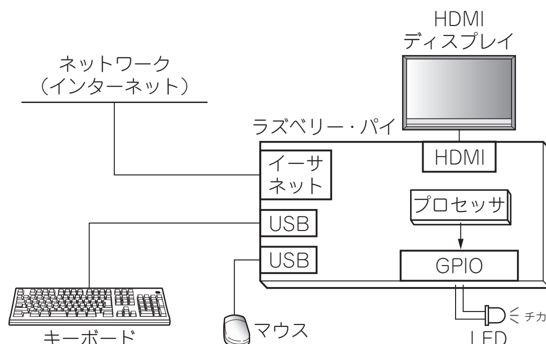


図2　シェルを使ったハードウェア操作のメカニズムをステップ・バイ・ステップで確かめてみる

UNIXハード制御の基本メカニズム

●ファイルの読み書きでデバイスを制御できる

UNIXでは仮想ファイルにデータを書き込むことによって，さまざまなデバイスの制御を行うことができます．UNIXは仮想ファイルへの書き込みをデバイスの操作と解釈し，内部でファイルに対応するデバイス・ドライバを呼び出し，対応する操作を行います（**図1**）．今回はLEDの点滅ということで，出力の例を示します．入力についても同じようにファイルを読み

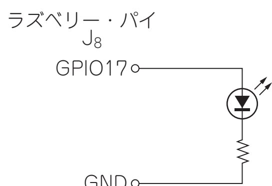


図3　実験回路

書きすることで操作できます．シェルはファイルを読み書きできるので，シェルを通じてデバイスを制御できます．シェルは対話的な環境なので，Cなどのコンパイルが必要な言語によって開発するよりも，はるかに容易に開発を行うことができます．

● ボード・コンピュータならではのユーザ権限

通常UNIXではこういったハードウェアを直接操作するようなファイルは，rootという特別なユーザしか操作できないようになっています．Raspbianでは，piユーザもこれらのファイルを操作できるよう，デフォルトで設定されています．

ステップ・バイ・ステップ！ シェルによるGPIO操作

ここでは，Raspbianが動作するラズベリー・パイ2 Model Bにおいて，シェルを通じてGPIOポートを操作する方法を説明します（**図2**）．GPIOの17番ピンにLEDを接続して，点滅させてみます．実験回路を**図3**に示します．

LEDを点灯/消灯する操作を**図4**に示します．UNIXでは仮想的なファイルを読み書きすることでさまざまな制御を行います．

● ステップ1：使用するピンを知らせる

GPIOの17番ピンを使用することをUNIXに知らせます（**図4**の①）．

```
echo 17>/sys/class/gpio/export
```

echoコマンドは引き数の文字列を標準出力に書き出すコマンドです．ここでは文字列“17”を，改行付きで標準出力に書き出します．

このコマンドでは，シェルのリダイレクト機能を使用しています．リダイレクト機能は入出力の先を変更する機能です．ここではechoコマンドの標準出力を，ファイル/sys/class/gpio/exportに変更しています．従ってこのコマンドは，ファイル/sys/class/gpio/exportに“17”という文字列を改行付きで書き込みます．すると，/sys/class/gpio/gpio17というファイルが作成されていることが確認できます．

● ステップ2：シンボリック・リンクの確認

/sys/class/gpio/gpio17は，シンボリック・リンクという種類のファイルです．

シンボリック・リンクとはファイル・ツリーの中のファイル，もしくはディレクトリを参照するファイルです．今回は，このファイルはディレクトリとして考えて構いません．

このディレクトリの中のファイルを操作すると，UNIXがGPIOの17番ピンの操作を行います．つまりシェルからGPIO17番ピンの操作ができるというわけです．

/sys/class/gpio/gpio17に含まれるファイルを確認します（**図4**の②）．

```
ls /sys/class/gpio/gpio17
```

表1のようなファイルがあること分かります．

```
pi@raspberrypi:~ $ ls -l /sys/class/gpio ←(ファイルを確認)
total 0
-rwxrwx--- 1 root gpio 4096 Aug  7 16:54 export
lrwxrwxrwx 1 root gpio    0 Jul  7 01:52 gpiochip0 -> ../../devices/platform/soc/3f200000.gpio/gpio/gpiochip0
lrwxrwxrwx 1 root gpio    0 Jul  7 01:52 gpiochip100 -> ../../devices/platform/soc/soc:virtgpio/gpio/
gpiochip100
-rwxrwx--- 1 root gpio 4096 Aug  7 16:55 unexport
pi@raspberrypi:~ $ echo 17 > /sys/class/gpio/export ←(①使用するピンを知らせる)
pi@raspberrypi:~ $ ls -l /sys/class/gpio ←(②gpio17というシンボリック・リンクができたことを確認)
total 0
-rwxrwx--- 1 root gpio 4096 Aug  7 16:55 export
lrwxrwxrwx 1 root gpio    0 Aug  7 16:55 gpio17 -> ../../devices/platform/soc/3f200000.gpio/gpio/gpio17
lrwxrwxrwx 1 root gpio    0 Jul  7 01:52 gpiochip0 -> ../../devices/platform/soc/3f200000.gpio/gpio/gpiochip0
lrwxrwxrwx 1 root gpio    0 Jul  7 01:52 gpiochip100 -> ../../devices/platform/soc/soc:virtgpio/gpio/
gpiochip100
-rwxrwx--- 1 root gpio 4096 Aug  7 16:55 unexport
pi@raspberrypi:~$ cat /sys/class/gpio/gpio17/direction ←(③GPIO17番ピンの入出力方向を確認)
in
pi@raspberrypi:~$ echo out > /sys/class/gpio/gpio17/direction ←(④GPIO17番ピンを出力方向に設定)
pi@raspberrypi:~$ echo 1 > /sys/class/gpio/gpio17/value ←(⑤LEDを点灯)
pi@raspberrypi:~$ echo 0 > /sys/class/gpio/gpio17/value ←(⑥LEDを消灯)
```

図4　LEDを点灯/消灯する操作

● ステップ3：ピンの入出力方向の確認

GPIOの17番ピンが入力ピンか出力ピンか調べてみます．シンボリック・リンクの中の，directionというファイルを読むと，inあるいはoutという出力が確認できます．

```
cat /sys/class/gpio/gpio17/direction
```

図4の③では入力を担っていることが分かります．

● ステップ4：出力ピンに設定

今回はLEDを点灯させるので，出力ピンに設定します（図4の④）．

```
echo out>/sys/class/gpio/gpio17/direction
```

ファイル/sys/class/gpio/gpio17/directionに文字列"out"を改行付きで書き込みます．UNIXがこれを解釈して，GPIOの17番ピンを出力ピンとして設定します．

● ステップ5：LEDを点灯

GPIOの17番ピンの出力を"H"にするか"L"にするかを切り替えるためには，/sys/class/gpio/gpio17/valueファイルを操作します．"H"にしたいときは文字列"1"，"L"にしたいときは文字列"0"を改行付きで書き込みます．

出力を"H"にしてLEDを点灯します（図4の⑤）．

```
echo 1>/sys/class/gpio/gpio17/value
```

/sys/class/gpio/gpio17/valueファイルに文字列"1"を改行付きで書き込みました．これでLEDが点灯します．もともと点灯してた場合は，変化しません．

● ステップ6：LEDを消灯

出力を"L"にしてLEDを消灯します（図4の⑥）．

表1　/sys/class/gpio/gpio17の中の主なファイル

ファイル名	機　能
direction	GPIO ピンの入出力の向き 読み込み：現在の向きの確認 書き込み：向きを設定
value	GPIO ピンの入出力の値 読み込み：入力値を読み込み 書き込み：GPIO ピンに出力
edge	GPIOピンの割り込みタイプ（none，falling，rising，bothの4種） 読み込み：割り込みタイプの確認 書き込み：割り込みタイプの設定
active_low	0以外の値を書き込む事でGPIOピンの入出力の向きを反転

```
echo 0>/sys/class/gpio/gpio17/value
```

/sys/class/gpio/gpio17/valueファイルに文字列"0"を改行付きで書き込みました．今度はLEDが消灯したはずです．このようにファイル/sys/class/gpio/gpio17/valueに値を書き込むことで，GPIOピンを操作できます．

● ステップ7：1秒おきに2回LEDを点滅

1秒おきに2回LEDを点滅させてみます（図5）．操作を図6に示します．そのためにまず，1秒おきに"1"と"0"を出力するコマンドを記述します（図6の①）．

```
{
  echo 1; sleep 1; echo 0; sleep 1;
  echo 1; sleep 1; echo 0; sleep 1;
}
```

最初は"$"というシェルのコマンド・プロンプトが表示されていますが，1行目を入力してEnterキーを押すとコマンド・プロンプトが">"に変わります．これは，前の行で入力したコマンドがまだ完結していないので，シェルが入力を促しているということです．

コマンドを実行すると，1秒おきに"1"と"0"が交互に1行ずつ，2回出力されます．これをファイル/sys/class/gpio/gpio17/valueに書き出せ

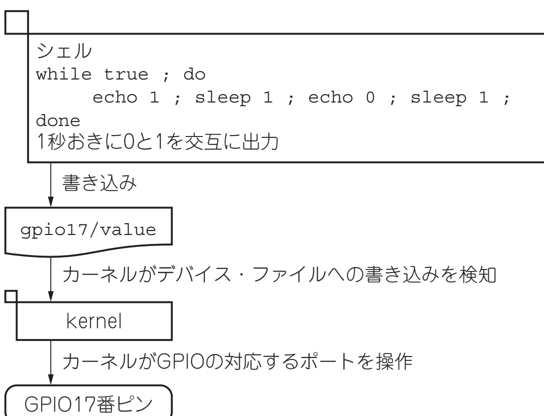


図5　シェル・スクリプトを使用したGPIOポート制御の仕組み

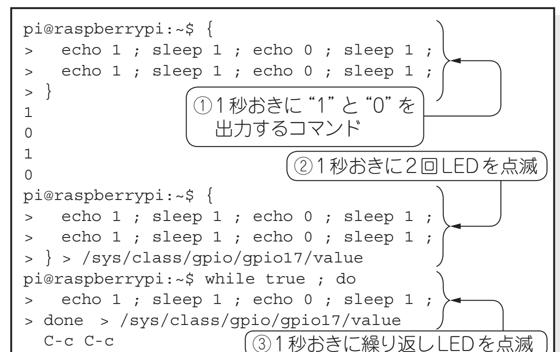


図6　LEDを点滅させる操作



図7 TwitterのWebAPIが提供されている小鳥男のウェブ・サイト…シェルと組み合わせて使ってみる

ば，1秒おきに2回LEDを点灯させることができます．

そこで1秒おきに“1”と“0”を出力するコマンドの出力をファイル/sys/class/gpio/gpio17/valueにリダイレクトします(**図6**の②)．

```
{
  echo 1; sleep 1; echo 0; sleep 1;
  echo 1; sleep 1; echo 0; sleep 1;
}     >/sys/class/gpio/gpio17/value
```

● ステップ8：1秒おきに繰り返しLEDを点滅

無限に1秒おきに点滅させてみます．シェルのwhile構文を用いた無限ループを使用して“1”と“0”の列を生成し，それをファイル/sys/class/gpio/gpio17/valueに書き込みます(**図6**の③)．

```
while true;do
  echo 1; sleep 1; echo 0; sleep 1;
done  >/sys/class/gpio/gpio17/value
```

ここで重要な点は，1秒おきに“1”と“0”が交互に1行ずつ出力する処理は，GPIOの17番ポートの制御とは何も関係がないということです．この処理は他のGPIOポートの制御や，あるいはファイルやターミナルに出力する場合でも同じ書き方になります．最後にファイル/sys/class/gpio/gpio17/valueに書き出すことによって，初めてGPIOの17番ポートの制御を行います．UNIXはデバイスの制御をファイルの読み書きという形に抽象化することによって，同じシェル・スクリプトをさまざまな状況で使用できるという，高度な部品化が可能になっています．

さすがシェル！他のソフトとの連携が簡単なことを確認する

シェルを通じてGPIOのデバイス・ファイルを操作して，LEDを点滅させることができました．シェルからは直接“1”と“0”を出力する以外にもさまざまなコマンドを呼び出して，パイプを利用してコマンドを組み合わせて使うことができます．

今度はTwitterクライアントの「小鳥男」(**図7**)を使って，ツイートが到着する度にLEDを点灯させるようにしてみます．

● 準備

小鳥男は以下の手順で準備します．手順を**図8**に示します．

```
sudo apt-get install git bc
git clone https://github.com/
ShellShoccar-jpn/kotoriotoko.git
PATH=/home/pi/kotoriotoko/APPS:/
home/pi/kotoriotoko/BIN:$PATH
```

1行目では必要なパッケージをインストールするコマンドを実行しています．gitパッケージは小鳥男が配布されているgithubから，小鳥男をダウンロードするためのコマンドのパッケージです．bcパッケージはRaspbianに標準でインストールされていない，POSIX標準コマンドのbcコマンドをインストールするためのコマンドです．

2行目で小鳥男をgithubからダウンロードしています．

3行目はシェルがコマンドを探索するファイル・パスに，小鳥男のディレクトリを追加しています．これでインストールが済みました．後は，TwitterのWebAPIを操作するための準備が必要です．文献(1)のマニュアルを参考にして準備を済ませておきます．

● ツイートが到着する度にLEDを点灯する

ツイートが到着する度にLEDを点灯する仕組みを**図9**に示します．ツイートが到着する度にLEDを点

```
pi@raspberrypi:~$ sudo apt-get install git bc
pi@raspberrypi:~$ #
pi@raspberrypi:~$ git clone https://github.com/ShellShoccar-jpn/kotoriotoko.git
fatal: destination path 'kotoriotoko' already exists and is not an empty directory.
pi@raspberrypi:~$ y
-bash: y: command not found
pi@raspberrypi:~$ PATH=/home/pi/kotoriotoko/APPS:/home/pi/kotoriotoko/BIN:$PATH
```

図8 TwitterのWebAPIが使える小鳥男の準備

```
pi@raspberrypi:~$ gathertw.sh -c -p2 電車 ← ①ツイートの取得
2016/08/08 03:15:09
- みく屋(旅) (@398eno)
- 電車とか乗ってて子供が騒ぐのは仕方ないことだから周りがどうこう言うことじゃないとは思うんだけど、親も声が大きくてうるさい上一切注意もしないで一緒に騒いでるのはガキがガキを育ててんじゃねーよって感じがします
- ret:0 fav:0
- https://twitter.com/398eno/status/762487285149425665
2016/08/08 03:15:08
 C-c C-c
pi@raspberrypi:~$ gathertw.sh -c -p2 電車          |  } ← ②httpsで始まる行だけを取り出す
> grep '^- https:'
- https://twitter.com/mikisato_/status/762487800692256770
- https://twitter.com/1dafullhiroya/status/762487797496242180
- https://twitter.com/hagarenkoi30/status/762487796611231744
 C-c C-c
pi@raspberrypi:~$ gathertw.sh -c -p2 電車          |
> grep '^- https:'                                 |
> while read ; do                                     } ← ③1行到着する度にLEDを点滅させる
>   echo 1 ; sleep 0.02 ; echo 0 ; sleep 0.02
> done > /sys/class/gpio/gpio17/value
2016/08/08-03:20:45 - 2016/08/08-03:19:48 gathered 100 tweet(s) (tot.100)
2016/08/08-03:19:46 - 2016/08/08-03:18:54 gathered 100 tweet(s) (tot.200)
  C-c C-c
```

図10　ツイートが到着する度にLEDを点灯させるための操作

灯させるための操作を**図10**に示します．

▶ステップ1：ツイートの取得

Twitterクライアントの準備ができているかどうかを確認するために，ツイートを取得してみます（**図10**の①）．

```
gathertw.sh -c -p2 電車
```

しばらく待つと，たくさんのツイートが次々と出力されてくると思います．これは，「電車」というキーワードで検索されたツイートです．そのままでは止まらないので，Crtl + Cキーで終了させます．

キーワードを変えるとまた違うツイートが検索されるので，いろいろ試してみると楽しいでしょう．

▶ステップ2：httpsで始まる行だけを取り出す

図10の①の出力では一つのツイートに関する情報が複数行に分かれて出力されていました．ここからツイートごとに1行で出力されるようにします．

ツイートの出力の中で，httpsで始まるURIが出力されている行があります．これは，それぞれのツイートを参照するためのURIで，ツイートごとに固有のものです．そこで，grepコマンドを使ってこの行のみを取り出します（**図10**の②）．

```
gathertw.sh -c -p2 電車          |
grep '^- https:'
```

▶ステップ3：1行到着する度にLEDを点滅させる

URIが行到着する度にLEDを点滅させます（**図10**の③）．

```
gathertw.sh -c -p2 電車          |
grep '^- https:'                 |
while read ; do
      echo 1 ; sleep 0.02 ; echo 0
; sleep 0.02
done > /sys/class/gpio/gpio17/value
```

3行目はシェルのwhile文です．中でreadというコマンドを呼んでいます．このコマンドを1行読むごとに終了します．そのため，while文の内容は，1行URIが到着する度に実行されます．

実行される内容はLEDの点滅実験の無限ループ内とほぼ同じですが，点滅の間隔を0.02秒にしています．LEDが猛烈な勢いで点滅することが確認できると思います．

このように，シェルを使うとUNIXの提供する機能を，それがたとえハードウェアであれ，WebAPIであれ，自在に組み合わせて新しい処理を記述できるのです．

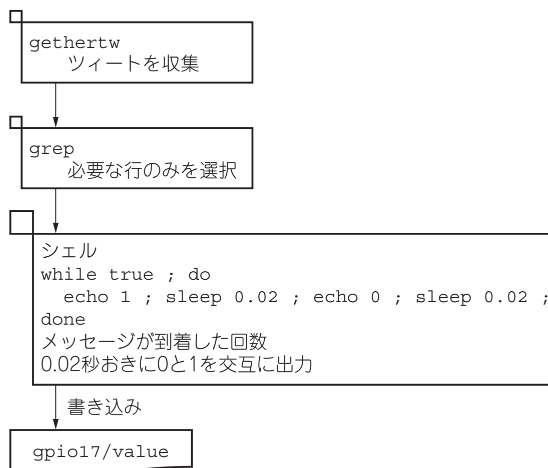


図9　小鳥男とGPIOの連携の仕組み

◆参考・引用*文献◆

(1) 小鳥男のマニュアル．
https://github.com/ShellShoccar-jpn/kotoriotoko

なかむら・かずたか

第2特集 第3章

コマンド&パイプはやっぱり強力! データ解析時代に欠かせない!

得意技①データ処理…IoT用データベース

中村 和敬

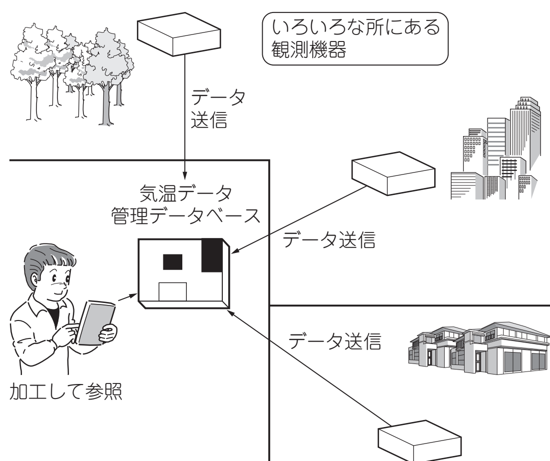


図1 IoT用データベースを作るときに想定した気温測定システム

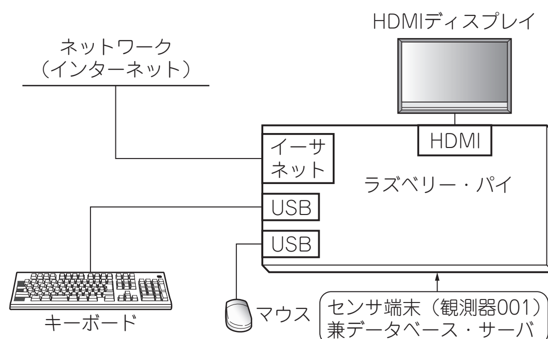


図2 得意技①:データ処理…IoT用シンプル・データベースを使ってみる
今回はセンサ端末とデータベース・サーバを1台のラズベリー・パイの中で仮想的に実現する

データベースはさまざまなデータを整理して保存し,必要に応じて取り出すことのできる仕組みです.その中でも関係代数に基づいたリレーショナル・データベースは,現代の多くの情報システムの基盤となっており,さまざまな専用のソフトウェアが開発されています.

シェルでもデータベースを作ることができます.データをファイルに格納してコマンドで操作することで,データを検索したり,付き合わせたり,加工したりするシステムが簡単に作成できます.

やること…シェルを使った IoT向けデータベースを作る

ここでは例として,さまざまな地点の気温の測定データを管理するデータベースを考えてみます(**図1**).さまざまな場所に配置されたIoTデバイスが気温を測定し,決まった時刻に観測データをデータベース・サーバに送信します.そして送られて来たデータをさまざまな形態に加工して利用します.

シェルでデータベース・システムを作る利点は,仕組みや使い方が容易に理解できるという点です.一般的なリレーショナル・データベースのアプリケーションを用いる場合には,仕組みや使い方を学ばなければなりません.しかしシェルであればそういった学習が不要で,シェルの知識をそのまま利用して処理を開発していくことができます.

特にシェルでは,パイプを利用して処理を段階的に記述していくことができるという特徴が,複雑な検索処理を記述する際に威力を発揮します.

他にも,管理するデータや処理の規模が大きくなっても,サーバの台数を増やすスケール・アウトによって対応することが容易だという利点もあります.シェルで構築したデータベースは,性能や要領などの増強が必要な場合でも,安価なハードウェアを追加することによって対応することができます.

● 実験に使うマシン…IoTも試せるラズパイ2

実験環境を**図2**に示します.

ハードウェアとしてコンパクトでIoTデバイスと接続しやすいため,ラズベリー・パイ2 Model Bを用います.piユーザでログインしており,ホーム・ディレクトリ直下は自由に操作できるとします.また,ラズベリー・パイで日本語が使用できるように設定されているという前提で,ファイル名などに日本語を使用し

```
pi@raspberrypi:~$ cd
pi@raspberrypi:~$ mkdir DB_SAMPLE
pi@raspberrypi:~$ cd DB_SAMPLE
pi@raspberrypi:~/DB_SAMPLE$ mkdir -p
DATA/INPUT DATA/INPUT.WORK DATA/
INPUT.DONE DATA/TABLE SHELL
pi@raspberrypi:~/DB_SAMPLE$ cat
<<EOF > DATA/TABLE/観測点
> 001 皇居   35.685175 139.752800
> 002 東京駅 35.681298 139.766247
> 003 永田町 35.676849 139.739919
> 004 霞ケ関 35.675277 139.752524
> EOF
pi@raspberrypi:~/DB_SAMPLE$ cat
<<EOF > DATA/TABLE/気温
> 20160823T060000 001 18
> 20160823T070000 001 19
> 20160823T080000 001 21
> 20160823T060000 002 24
> 20160823T070000 002 26
> 20160823T080000 002 27
> 20160823T060000 003 22
> 20160823T070000 003 23
> 20160823T080000 003 24
> 20160823T060000 004 21
> 20160823T070000 004 23
> 20160823T080000 004 24
> EOF
```

①作業ディレクトリの作成
②観測点データのファイルを作成
③気温の時系列データのファイルを作成

図3　実験の準備

```
DB_SAMPLE/：データベース・アプリケーションを配置するディレクトリ
├──DATA：データを配置するディレクトリ
│ ├──INPUT：入力ファイルを配置するディレクトリ
│ ├──INPUT. WORK：処理中の入力ファイルを配置するディレクトリ
│ ├──INPUT. DONE：処理済みの入力ファイルを配置するディレクトリ
│ └──TABLE：整理したデータを配置するディレクトリ
└──SHELL：データを処理するファイルを配置するディレクトリ
```

図4　作業ディレクトリの作成

ます．

実験用サンプル・データの準備

準備の手順を図3に示します．

● ステップ1：作業ディレクトリの作成

データやスクリプトを格納するディレクトリを作成します（図3の①）．

```
cd
mkdir DB_SAMPLE
cd DB_SAMPLE
mkdir -p DATA/INPUT DATA/INPUT.WORK
DATA/INPUT.DONE DATA/TABLE SHELL
```

ホーム・ディレクトリにDB_SAMPLEというディレクトリができて，DB_SAMPLEの中に図4のようなディレクトリ・ツリーができます．

● ステップ2：サンプル・データの作成

実験で使用するデータを作成します．データは観測点に関するテーブルと，それぞれの観測点での気温の時系列データのテーブルの2種類を作ります．それぞれ表1と表2のような形になります．

表1　観測点に関するデータ

観測点番号	名　前	緯　度	経　度
001	皇居	35.685175	139.752800
002	東京駅	35.681298	139.766247
003	永田町	35.676849	139.739919
004	霞ケ関	35.675277	139.752524

表2　観測点での気温の時系列データ

日　時	観測点番号	気　温
20160823T060000	001	18
20160823T060000	002	24
20160823T060000	003	22
20160823T060000	004	21
20160823T070000	001	19
20160823T070000	002	26
20160823T070000	003	23
20160823T070000	004	23

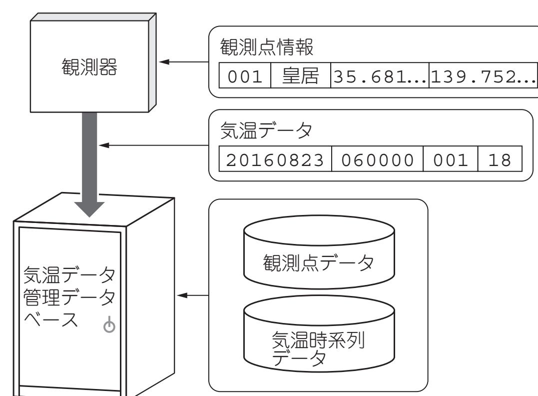


図5　管理するデータと実際のノードの関係

▶観測点データ

観測点データは，観測点を識別するための番号と，観測点についての情報からなっています．気温の時系列データは，観測時刻と観測点の番号，気温からなっています（図5）．

観測時刻については，ISO 8601形式で表記しています．

これらのテーブルをファイルに格納する場合，行指向スペース区切りのテキスト・データとして保存します．まずは観測点データのファイルを作成してみます（図3の②）注1．

注1：以降，特に指定しない限り，同じ記事中の前のコマンドに続けてコマンドが入力されるものと仮定する．

```
pi@raspberrypi:~/DB_SAMPLE$ awk '$1 == "001"'
DATA/TABLE/観測点
001 皇居   35.685175 139.752800
pi@raspberrypi:~/DB_SAMPLE$ awk
'"20160823T060000" <= $1 && $1 <=
"20160823T075959"' DATA/TABLE/気温
20160823T060000 001 18
20160823T070000 001 19
20160823T060000 002 24
20160823T070000 002 26
20160823T060000 003 22
20160823T070000 003 23
20160823T060000 004 21
20160823T070000 004 23
```


図6　データベースの基本操作①ー検索

```
cat <<EOF > DATA/TABLE/観測点
001 皇居   35.685175 139.752800
002 東京駅 35.681298 139.766247
003 永田町 35.676849 139.739919
004 霞ケ関 35.675277 139.752524
EOF
```

このコマンドを実行するとcatコマンドに対する標準入力の内容が標準出力に書き出されます．<<EOFはシェルの機能の一つで，ヒア・ドキュメント機能を利用するための記述です．このように記述すると，起動したコマンドの標準入力に，それ以降の行が渡され，“<<”に続くキーワードが入力されると標準入力を閉じます．標準出力の内容は“>”以降で指定されたファイルに書き出されます．ここでは，“DATA/TABLE/観測点”です．

```
cat DATA/TABLE/観測点
```

と入力するとこのファイルの内容が表示されます．入力した表と同じ内容が(EOFを除いて)表示されれば成功です．

```
awk '"20160823T060000" <= $1 &&
                         $1 <= "20160823T075959"'
```


図7　awkコマンドによる絞り込み処理

▶気温の時系列データ

気温の時系列データのファイルを作成します(図3の③)．

```
cat <<EOF > DATA/TABLE/気温
20160823T060000 001 18
20160823T070000 001 19
…
20160823T080000 004 24
EOF
```

観測点データと同じように，DATA/TABLE/気温に入力内容が格納されます．

IoT用データベースの基本機能

● 検索

作成したデータから，必要なデータを検索します．操作の例を図6に示します．

例えば観測点001番のデータを取り出したい場合には，図6の①の操作を行います．

```
awk '$1 == "001"' DATA/TABLE/観測点
```

観測点001番のレコードが表示されます．

```
001 皇居   35.685175 139.752800
```

ここで使用したのはawkというコマンドです．行指向のテキスト・データを操作するための汎用コマンドであり，さまざまな処理を完結に記述することができます．このコマンドは，第1フィールドが“001”である行を表示せよという処理になります．“$”に続く数値を変えたり，比較演算子を使用することでざまざまな検索が可能です．

例えば図6の②のコマンドは，2016年8月23日の6時から7時の間の観測データを表示せよ，という処理になります(図7)．

```
awk '"20160823T060000" <= $1 &&
$1 <= "20160823T075959"' DATA/TABLE
/気温.201608
```

以下のような出力が得られます．

```
20160823T060000 001 18
20160823T070000 001 19
…
20160823T070000 004 23
```

● ソート

今回作成した気温ファイルは，観測点→観測日時の順に整列されていました．これを観測日時→観測点の順に並べ替えてみます．操作の例を図8に示します．

```
sort -k1,2 DATA/TABLE/気温
```

ここではsortコマンドを使用します(図9)．図8の①のコマンドでは観測日時ごとに整列して出力されます．

```
pi@raspberrypi:~/DB_SAMPLE$ sort -k1,2 DATA/TABLE/気温
20160823T060000 001 18
20160823T060000 002 24
20160823T060000 003 22
20160823T060000 004 21
20160823T070000 001 19
20160823T070000 002 26
20160823T070000 003 23
20160823T070000 004 23
20160823T080000 001 21
20160823T080000 002 27
20160823T080000 003 24
20160823T080000 004 24
pi@raspberrypi:~/DB_SAMPLE$ sort -k3,3n DATA/TABLE/気温
20160823T060000 001 18
20160823T070000 001 19
20160823T060000 004 21
20160823T080000 001 21
20160823T060000 003 22
20160823T070000 003 23
20160823T070000 004 23
20160823T060000 002 24
20160823T080000 003 24
20160823T080000 004 24
20160823T070000 002 26
20160823T080000 002 27
```


図8　データベースの基本操作②ーソート

```
sort -k1,2 DATA/TABLE/気温
```


ソート前	ソート後
20160823T060000 001 18	20160823T060000 001 18
20160823T070000 001 19	20160823T070000 001 19
20160823T080000 001 21	20160823T060000 004 21
20160823T060000 002 24	20160823T080000 001 21
20160823T070000 002 26	20160823T060000 003 22
20160823T080000 002 27	20160823T070000 003 23
20160823T060000 003 22	20160823T070000 004 23
20160823T070000 003 23	20160823T060000 002 24
20160823T080000 003 24	20160823T080000 003 24
20160823T060000 004 21	20160823T080000 004 24
20160823T070000 004 23	20160823T070000 002 26
20160823T080000 004 24	20160823T080000 002 27

図9　sortコマンドによるソート処理

```
20160823T060000 001 18
20160823T060000 002 24
…
20160823T080000 004 24
```

図8の②のように操作すると，気温の低い順にデータを整列させることができます．

```
sort -k3,3n DATA/TABLE/気温
```

以下のような出力が得られます．

```
20160823T060000 001 18
20160823T070000 001 19
…
20160823T080000 002 27
```

検索とソート，そして先頭や末尾のレコードのみを出力するheadやtailなどのコマンドを使用することで，さまざまな検索が可能です．

● データの突き合わせ

気温ファイルには，観測点については観測点の番号しか記載がありません．そこで，データに観測点についての情報を付加してみます．操作の例を**図10**に示します．

```
join -1 1 -2 2 DATA/TABLE/観測点 DATA/TABLE/気温
```

ここではjoinコマンドを使用しています（**図11**）．

```
pi@raspberrypi:~/DB_SAMPLE$ join -1 1 -2 2 DATA/
TABLE/観測点 DATA/TABLE/気温
001 皇居 35.685175 139.752800 20160823T060000 18
001 皇居 35.685175 139.752800 20160823T070000 19
001 皇居 35.685175 139.752800 20160823T080000 21
002 東京駅 35.681298 139.766247 20160823T060000 24
002 東京駅 35.681298 139.766247 20160823T070000 26
002 東京駅 35.681298 139.766247 20160823T080000 27
003 永田町 35.676849 139.739919 20160823T060000 22
003 永田町 35.676849 139.739919 20160823T070000 23
003 永田町 35.676849 139.739919 20160823T080000 24
004 霞ケ関 35.675277 139.752524 20160823T060000 21
004 霞ケ関 35.675277 139.752524 20160823T070000 23
004 霞ケ関 35.675277 139.752524 20160823T080000 24
```

図10　データベースの基本操作③ーデータの突き合わせ

テーブル1

001	皇居	35.685175	139.752800
002	東京駅	35.681298	139.766247
003	永田町	35.676849	139.739919
004	霞ケ関	35.675277	139.752524

テーブル2

20160823T060000	001	18
20160823T070000	001	19
20160823T080000	001	21
20160823T060000	002	24
20160823T070000	002	26
20160823T080000	002	27
20160823T060000	003	22
20160823T070000	003	23
20160823T080000	003	24
20160823T060000	004	21
20160823T070000	004	23
20160823T080000	004	24

キー	テーブル1由来	テーブル2由来
001	皇居 35.685175 139.752800	20160823T060000 18
001	皇居 35.685175 139.752800	20160823T070000 19
001	皇居 35.685175 139.752800	20160823T080000 21
002	東京駅 35.681298 139.766247	20160823T060000 24
002	東京駅 35.681298 139.766247	20160823T070000 26
002	東京駅 35.681298 139.766247	20160823T080000 27
003	永田町 35.676849 139.739919	20160823T060000 22
003	永田町 35.676849 139.739919	20160823T070000 23
003	永田町 35.676849 139.739919	20160823T080000 24
004	霞ケ関 35.675277 139.752524	20160823T060000 21
004	霞ケ関 35.675277 139.752524	20160823T070000 23
004	霞ケ関 35.675277 139.752524	20160823T080000 24

図11　joinコマンドによるデータの突き合わせ処理

```
pi@raspberrypi:~/DB_SAMPLE$ sort DATA/TABLE/気温 |
> join -1 1 -2 2 DATA/TABLE/観測点 -
001 皇居 35.685175 139.752800 20160823T060000 18
002 東京駅 35.681298 139.766247 20160823T060000 24
003 永田町 35.676849 139.739919 20160823T060000 22
004 霞ヶ関 35.675277 139.752524 20160823T060000 21
```

図12 結合の前にはソートが必要

ファイル1のDATA/TABLE/観測点の第1フィールドと，ファイル2のDATA/TABLE/気温の第2フィールドを突き合わせよという意味です.

出力は以下のようになります.

```
001  皇 居 35.685175 139.752800
20160823T060000 18
001  皇 居 35.685175 139.752800
20160823T070000 19
…
004  霞 ヶ 関 35.675277 139.752524
20160823T080000 24
```

観測点番号で，DATA/TABLE/観測点とDATA/TABLE/気温の二つのファイルが連結され，第1フィールドから，観測点番号，観測点名，緯度，経度，観測日時，気温の順に，データが出力されました.

● 注意！結合の前にはソートが必要

ここで使用したファイルDATA/TABLE/観測点とDATA/TABLE/気温は，結合に使用するデータで整列されている必要があります.

例えば，観測日時→観測点の順で整列されたデータに，観測点の情報を付加するために，**図12**の操作を行ったとしましょう.

```
sort DATA/TABLE/気温  |
join -1 1 -2 2 DATA/TABLE/観測点 -
```

出力は以下のようになります.

```
001  皇 居 35.685175 139.752800
20160823T060000 18
002  東 京 駅 35.681298 139.766247
20160823T060000 24
003  永 田 町 35.676849 139.739919
20160823T060000 22
004  霞 ヶ 関 35.675277 139.752524
20160823T060000 21
```

これは，1行目のsortコマンドにより，気温データが観測日時→観測点の順で整列されてしまったからです．2行目ではjoinコマンドのファイル2に“-”，つまり標準入力を指定しており，前段で整列された気温データが入力されます．先ほどとは違い，観測点番号で整列されていないので，このような出力になってしまいました.

joinコマンドによるファイルの突き合わせの際には，レコードの並び順に注意が必要です．そのため，データをファイルに格納する時点で，レコードを適切に整列させて格納しておきましょう.

● 複雑なデータ処理もシンプルに記述できる

突き合わせ処理は，検索処理と組み合わせによって，より複雑な処理を行うこともできます．例えば，緯度139.75～139.76の間の観測点の，2016年8月23日の6時から7時の観測データを出力してみます(**図13**)．操作を**図14**に示します.

```
awk '139.75 <= $4 && $4 <= 139.76'
DATA/TABLE/観測点  |
join -1 1 -2 2 - DATA/TABLE/気温  |
awk '"20160823T060000" <= $5 && $5
<= "20160823T075959"'
```

1行目で緯度139.75～139.76の間の観測点のレコードを抽出し，2行目で絞り込んだ観測点データと気温データを突き合わせて，3行目で時刻の絞り込みを行っています.

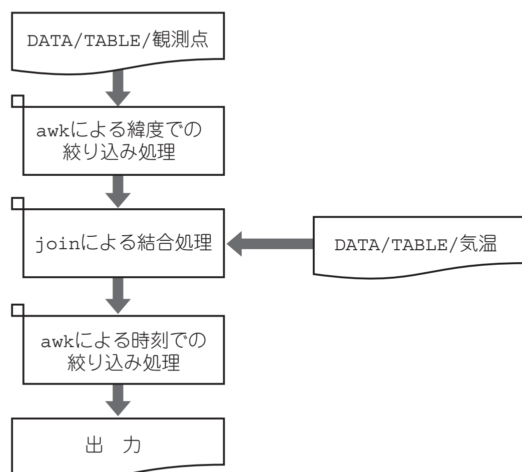


図13 検索処理と突き合わせ処理の組み合わせ

```
pi@raspberrypi:~/DB_SAMPLE$ awk '139.75 <= $4 && $4
<= 139.76' DATA/TABLE/観測点  |
> join -1 1 -2 2 - DATA/TABLE/気温  |
> awk '"20160823T060000" <= $5 && $5 <=
"20160823T075959"'
001 皇居 35.685175 139.752800 20160823T060000 18
001 皇居 35.685175 139.752800 20160823T070000 19
004 霞ヶ関 35.675277 139.752524 20160823T060000 21
004 霞ヶ関 35.675277 139.752524 20160823T070000 23
```

図14 検索処理と突き合わせ処理の組み合わせて緯度139.75～139.76の間の観測点の，2016年8月23日の6時から7時の観測データを出力

joinコマンドの出力データ

キー	テーブル1由来			テーブル2由来	
001	皇　居	35.685175	139.752800	20160823T060000	18
001	皇　居	35.685175	139.752800	20160823T070000	19
001	皇　居	35.685175	139.752800	20160823T080000	21
002	東京駅	35.681298	139.766247	20160823T060000	24
002	東京駅	35.681298	139.766247	20160823T070000	26
002	東京駅	35.681298	139.766247	20160823T080000	27
003	永田町	35.676849	139.739919	20160823T060000	22
003	永田町	35.676849	139.739919	20160823T070000	23
003	永田町	35.676849	139.739919	20160823T080000	24
004	霞ヶ関	35.675277	139.752524	20160823T060000	21
004	霞ヶ関	35.675277	139.752524	20160823T070000	23
004	霞ヶ関	35.675277	139.752524	20160823T080000	24



20160823T060000	001	皇居	18
20160823T070000	001	皇居	19
20160823T080000	001	皇居	21
20160823T060000	002	東京駅	24
20160823T070000	002	東京駅	26
20160823T080000	002	東京駅	27
20160823T060000	003	永田町	22
20160823T070000	003	永田町	23
20160823T080000	003	永田町	24
20160823T060000	004	霞ヶ関	21
20160823T070000	004	霞ヶ関	23
20160823T080000	004	霞ヶ関	24

図16　フィールド並べ替え処理

```
pi@raspberrypi:~/DB_SAMPLE$ join -1 1 -2 2 DATA/
TABLE/観測点 DATA/TABLE/気温  |
> awk '{ print $5, $1, $2, $6 }'
20160823T060000 001 皇居 18
20160823T070000 001 皇居 19
20160823T080000 001 皇居 21
20160823T060000 002 東京駅 24
20160823T070000 002 東京駅 26
20160823T080000 002 東京駅 27
20160823T060000 003 永田町 22
20160823T070000 003 永田町 23
20160823T080000 003 永田町 24
20160823T060000 004 霞ヶ関 21
20160823T070000 004 霞ヶ関 23
20160823T080000 004 霞ヶ関 24
```

図15　データベースの編集操作①ーフィールドの並べ直し

この処理の出力は以下のようになります．

```
001　皇 居 35.685175 139.752800
20160823T060000 18
001　皇 居 35.685175 139.752800
20160823T070000 19
004　霞 ヶ 関 35.675277 139.752524
20160823T060000 21
004　霞 ヶ 関 35.675277 139.752524
20160823T070000 23
```

このようにパイプで処理を次々と結合していくのが，シェル流のプログラミングです．1行目の出力はパイプ（|）によって2行目の入力となり，2行目の出力はまたパイプによって3行目の入力となっています．分かりにくい場合は1段ずつ処理を追加して実行してみましょう．

機能1：データの編集

データを編集します．ここではawkコマンドを使用します．

● フィールドの並べ直し

joinコマンドで結合したデータは，第1フィールドから，観測点番号，観測点名，緯度，経度，観測日時，気温の順に並んでいました．今度はこれを，観測日時，観測点番号，観測点名，気温の順に並び直してみます（**図15**）．

```
join -1 1 -2 2 DATA/TABLE/観測点
DATA/TABLE/気温  |
awk '{ print $5, $1, $2, $6 }'
```

1行目はjoinコマンドを利用した突き合わせです．その出力を2行目のawkコマンドで並べ替えて，第5フィールド，第1フィールド，第2フィールド，第6フィールドの順に出力しています（**図16**）．

このコマンドの出力は以下のようになります．

```
20160823T060000 001 皇居 18
20160823T070000 001 皇居 19
…
20160823T080000 004 霞ヶ関 24
```

```
pi@raspberrypi:~/DB_SAMPLE$ join -1 1 -2 2 DATA/
TABLE/観測点 DATA/TABLE/気温  |
> awk '!("20160823T060000" <= $5 && $5 <=
"20160823T075959"){ print $5, $1, $2, $6 }
>        "20160823T060000" <= $5 && $5 <=
"20160823T075959" { print $5, $1, $2, $6, "*" }'
20160823T060000 001 皇居 18 *
20160823T070000 001 皇居 19 *
20160823T080000 001 皇居 21
20160823T060000 002 東京駅 24 *
20160823T070000 002 東京駅 26 *
20160823T080000 002 東京駅 27
20160823T060000 003 永田町 22 *
20160823T070000 003 永田町 23 *
20160823T080000 003 永田町 24
20160823T060000 004 霞ケ関 21 *
20160823T070000 004 霞ケ関 23 *
20160823T080000 004 霞ケ関 24
```

図17 データベースの編集操作②ー文字の追加

実はこのような並べ替えはjoinコマンドのオプションを駆使することでも可能です．しかし長期的な保守性の観点からは，joinコマンドはあくまでもデータの突き合わせのコマンドとして使用し，データの編集にはawkという形で使い分けるのが望ましいでしょう．

● 文字の追加

awkコマンドは本格的なプログラム言語処理系ですので，さまざまな処理を記述することができます．プログラム言語AWKは，入力1行ごとにどのような処理するかを，「パターン」と「アクション」の組み合わせで記述します．そして，パターンにマッチした行についてアクションを実行します．パターンとアクションの組み合わせは，複数記述可能です．

2016年8月23日の6時から7時の観測データには，最後のフィールドに“*”記号を付加するようにします(**図17**)．

```
join -1 1 -2 2 DATA/TABLE/観測点
DATA/TABLE/気温          |
awk '!("20160823T060000" <= $5 &&
$5 <= "20160823T075959"){ print $5,
$1, $2, $6 }
        "20160823T060000" <= $5 &&
$5 <= "20160823T075959" { print $5,
$1, $2, $6, "*" }'
```

3行目は，「!("20160823T060000" <= $5 && $5 <= "20160823T075959")」がパターンであり，「{ print $5, $1, $2, $6 }」がアクションです(**図18**)．2016年8月23日の6時から7時の観測データではない行はこのパターンにマッチし，先ほどと同じアクションが実行されます．

4行目は「("20160823T060000" <= $5 && $5 <= "20160823T075959")」がパターンであり，「{ print $5, $1, $2, $6, "*" }」がアクションです．2016年8月23日の6時から7時の観測データの行はこのパターンにマッチし，最後のフィールドに“*”記号を付加して出力します．パターンのみが記述され，アクションが省略された場合は，パターンにマッチした行をそのまま出力します．パターンが省略され，アクションのみが記述された場合は，全ての行についてそのアクションを実行します．

● 観測点ごとに1レコードにまとめて出力

awkによる編集は，検索処理や突き合わせ処理，組み合わせを使った，より複雑な処理を行うこともできます．突き合わせの結果を，観測点ごとに1レコードで出力してみます(**図19**)．

1～3行目は突き合わせ処理のままです．4～6行目が，観測点ごとに1レコードにまとめて出力するための処理です(**図20**)．

keyという変数には観測点番号が代入されます．読み込まれたデータの観測点番号がkeyに代入されている観測点番号と同じである場合，読み込まれたデータは改行なしで行末にスペースを追加して出力されます．格納されている観測点番号と異なる観測点番号のデータが読み込まれると，新しい観測点番号をkeyに代入して，行頭に改行を付けて行末にスペースを追加して出力されます．最後のENDと書かれた

図18 awkコマンドは「パターン」と「アクション」の組み合わせで記述する

```
pi@raspberrypi:~/DB_SAMPLE$ awk '139.75 <= $4 && $4 <= 139.76' DATA/TABLE/観測点  |
> join -1 1 -2 2 - DATA/TABLE/気温  |
> awk '"20160823T060000" <= $5 && $5 <= "20160823T075959"'|
> awk '    key==$1{ printf("%s ", $0) }
>   key!=$1{ key=$1 ; printf("¥n%s ", $0); }
>   END{ printf("¥n") }'  |
> awk 'NR!=1'
001 皇居 35.685175 139.752800 20160823T060000 18 001 皇居 35.685175 139.752800 20160823T070000 19
004 霞ヶ関 35.675277 139.752524 20160823T060000 21 004 霞ヶ関 35.675277 139.752524 20160823T070000 23
```

図19　データベースの編集操作③－1レコードをまとめる

パターンは，ファイルを読み終えた後に実行されるパターンです．最後に改行文字を出力します．

このような書き方をした場合，1行目に空の行が出力されてしまいます．7行目はそのような行を削除するための処理です．NRは特殊な変数で，そのとき読み込まれている行が何行目かが代入されています．ここではNR!=1つまり，1行目でない場合にマッチする行が出力されます．

出力は8～9行目のようになります．

今回は3～7行目，awkを3段階に分けて呼び出しました．これは1段階のみでも記述することが可能です．しかし，シェルの場合このように自分の分かりやすいように段階を踏んで処理を記述していけるので，素早く必要な処理を記述することができます．

AWKは算術演算も可能なので，平均気温の計算や分散の計算なども可能です．

機能2：データの送受信

気温のデータベースであれば，次々と新しい観測データを追加していきたいところです．そのためにまず，どのようにして観測データをデータベースに送信/受信すればよいか考えてみます．

● 準備

準備の手順を図21に示します．

データベースも各観測地点の観測器も，ラズベリー・パイで実装されており，インターネットで通信が可能だとします．今回は簡単のため，観測点番号001番の観測器は，データベース・サーバも兼ねているとします．

温度はtemperatureというコマンドを実行すると，標準出力に現在の気温が出力されるとします(図21の②)．

```
temperature
23
```

今回このようなコマンドはないので，図21の①のように，常に23℃を返すシェル・スクリプトを作成します．

```
echo "echo 23" > SHELL/temperture
chmod +x SHELL/temperture
```

1行目でecho 23というコマンドを呼び出すファイル，SHELL/tempertureを作成しています．

2行目は今作成したファイルSHELL/tempertureをシェルからコマンドとして実行できるようにするため，chmodコマンドを利用して，ファイルSHELL/tempertureを実行可能に変更しています．

ラズベリー・パイ間の通信にはsshコマンドを利

awk
処理1
パターン	アクション	
key==$1	{ printf("%s ", $0) }	パターン　：keyとキー(第1フィールド)が同じなら アクション：1. 行末の改行なしで出力

処理2
パターン	アクション	
key!=$1	{ key=$1 ; printf("¥n%s ", $0); }	パターン　：keyとキー(第1フィールド)が異なるなら アクション：1. キーを更新して 2. 行頭に改行を付けて行末の改行なしで出力

処理3
パターン	アクション	
END	{ printf("¥n") }	パターン　：最終行を出力し終えたら アクション：改行を出力

図20　観測点ごとに1レコードにまとめて出力するawkコマンドの構造

```
pi@raspberrypi:~/DB_SAMPLE$ echo "echo 23" > SHELL/temperture
pi@raspberrypi:~/DB_SAMPLE$ chmod +x SHELL/temperture
pi@raspberrypi:~/DB_SAMPLE$ SHELL/temperture
23
pi@raspberrypi:~/DB_SAMPLE$ ssh-keygen
-bash: ssh-keyge: command not found
Generating public/private rsa key pair.
Enter file in which to save the key (/home/pi/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/pi/.ssh/id_rsa.
Your public key has been saved in /home/pi/.ssh/id_rsa.pub.
The key fingerprint is:
b3:b9:e9:5b:cd:f4:46:12:11:c3:52:f3:b1:82:79:59 pi@raspberrypi
The key's randomart image is:
+---[RSA 2048]----+
|          o*.E   |
|         .o.B o  |
|         o.= o   |
|           . o   |
|       S    o .  |
|        + + +    |
|       o . o o   |
|        +    .   |
|       .=.       |
+-----------------+
pi@raspberrypi:~/DB_SAMPLE$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

①常に23℃を返すコマンドの作成
②作成したコマンドの確認
③sshコマンドを使えるようにする

図21　データの送受信の準備

用します．sshコマンドをスクリプト中で使用するために図21の③のコマンドを実行します．

```
ssh-keygen
[コマンドプロンプトが表示されるまでエンターキーを連打]
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

このようにすると，ローカル・ホストと通信する際に公開鍵認証という方法を使用することができるようになります．パスワードを入力せずにローカル・ホストにログインできるようにしておきます．

● 観測器からサーバへの送信

準備ができた状況で，観測点番号001番の観測器からデータベース・サーバにデータを送る場合を考えてみます（図22）．

```
time=$(date +%Y%m%dT%H%M%S)
opid="001"
echo "$time $opid $(~/DB_SAMPLE/SHELL/temperture)" |
ssh pi@localhost "cat > ~/DB_SAMPLE/DATA/INPUT/DATA.$opid.$time.$$"
```

1行目で計測時刻を取得してシェル変数timeに格納し，2行目でシェル変数opidに観測点番号を格納しています．3行目でechoコマンドを使用して，温度時系列データのレコードを生成しています．「$(~/DB_SAMPLE/SHELL/temperture)」はシェルのコマンド置換という機能を使用しており，SHELL/temperture コマンドの実行結果がこの文字列と置き換えられます．

4行目では，sshコマンドを使用してデータベース・サーバに接続してコマンドを実行しています．localhostは，データベース・サーバのIPアドレス，またはホスト名です．今回は観測点番号001番の観測器と同じなので，自分自身を示すホスト名localhostが入っています．そして，「cat > ~/DB_SAMPLE/DATA/INPUT/DATA.$opid.$time.$$」がデータベース・サーバで実行されるコマンドです．これは標準入力から読み込んだ内容を，~/DB_SAMPLE/DATA/INPUT/DATA.$opid.$time.$$に書き込むという処理です．そのため，3行目で観測点番号001番の観測器で生成された温度時系列データのレコードが，データベース・サーバの~/DB_SAMPLE/DATA/INPUT/DATA.$opid.$time.$$というファイルに格納されます．今回はどちらも同じサーバ

```
pi@raspberrypi:~/DB_SAMPLE$ time=$(date +%Y%m%dT%H%M%S)
pi@raspberrypi:~/DB_SAMPLE$ opid="001"
pi@raspberrypi:~/DB_SAMPLE$ echo "$time $opid $(~/DB_SAMPLE/SHELL/temperture)"  |
> ssh pi@localhost "cat > ~/DB_SAMPLE/DATA/INPUT/DATA.$opid.$time.$$"
```

図22　001番の観測器からデータベース・サーバにデータを送る

なので，同じサーバにファイルが作成されます．

このファイル名で注意が必要なのは，末尾の$opid.$time.$$という部分です．このうち，$opidは先ほど代入した観測点番号のシェル変数opid，$timeは計測時刻のシェル変数timeに展開されます．$$は，この処理を実行したシェルのプロセス識別番号（プロセスID）に展開されます．プロセス識別番号は1秒程度の短い間に同じ番号が再利用されることはありません．そのため，この三つを連結してファイルの末尾に付加することで，ユニークなファイル名を作成し，他のタイミングで送られたり，他の観測点の観測器から送られたデータとファイル名がかぶったりしないようにしています．

● サーバに作成されたファイルの確認

作成されたファイルを確認します（図23）．

```
head DATA/INPUT/*
```

すると以下のような出力が得られます．

```
20160823T113224 001 23
```

第1フィールドの値には，実際にはコマンドを実行した日時が入ります．もし複数回上述のコマンドを実行していた場合には，以下のような出力になるでしょう．

```
==>
DATA/INPUT/DATA.001.20160823T113224.
2268 <==
20160823T113224 001 23

==>
DATA/INPUT/DATA.001.20160823T113356.
2275 <==
20160823T113356 001 23
```

headコマンドは，指定されたファイルの先頭行を何行か出力するコマンドです．今回はシェルのワイルドカード機能を使用してファイルを指定しているので，シェルはDATA/INPUT/*にマッチする全てのファイル・パスをheadコマンドに引き渡します．複数のファイルが指定された場合には，ファイルごとにファイル・パスを==>と<==でくくって出力してくれます．コマンドを複数回実行した場合ファイルが複数作成されます．2番目の出力例は，作成されたファイルは二つで，1行目と4行目がファイル・パスで2行目と5行目がそれぞれの内容になっています．headコマンドはファイルの内容を簡単に確認したい場合には便利です．

```
pi@raspberrypi:~/DB_SAMPLE$ head DATA/INPUT/*
==> DATA/INPUT/DATA.001.20160823T113224.2268 <==
20160823T113224 001 23

==> DATA/INPUT/DATA.001.20160823T113224.2268 <==
20160823T113224 001 23

==> DATA/INPUT/DATA.001.20160823T113356.2275 <==
20160823T113356 001 23
```

図23　データベース・サーバに作成されたファイルの確認

● 1時間おきに測定データをサーバに登録する

今回は1時間おきに各観測器で温度を測定して，データベース・サーバに登録してみます（図24）．ここでは，cronというUNIX標準のデーモンを使用するのが簡単です．

▶ステップ1：観測器からの送信コマンドの作成

観測器からサーバへの送信処理（図22）をシェル・スクリプトにします．

```
cat <<'EOS' > SHELL/SEND_TEMPERTURE
time=$(date +%Y%m%dT%H%M%S)
opid=$1
```

```
pi@raspberrypi:~/DB_SAMPLE$ cat <<'EOS' > SHELL/SEND_TEMPERTURE   ←①観測器からサーバへの送信処理をシェル・スクリプトにする
> time=$(date +%Y%m%dT%H%M%S)
> opid=$1
> echo "$time $opid $(~/DB_SAMPLE/SHELL/temperture)"     |
> ssh pi@localhost "cat > ~/DB_SAMPLE/DATA/INPUT/DATA.$opid.$time.$$"
> EOS
pi@raspberrypi:~/DB_SAMPLE$ chmod +x SHELL/SEND_TEMPERTURE
pi@raspberrypi:~/DB_SAMPLE$ select-editor   ←②-1エディタの起動
Select an editor.  To change later, run 'select-editor'.
  1. /bin/ed
  2. /bin/nano        <---- easiest
  3. /usr/bin/vim.basic
  4. /usr/bin/vim.tiny

Choose 1-4 [2]: 4
pi@raspberrypi:~/DB_SAMPLE$ crontab -e


②-2エディタが起動する．以下の通りに入力する
 0 *    *   *   *   ~/DB_SAMPLE/SHELL/SEND_TEMPERTURE 001
```

図24　1時間おきに測定データをサーバに登録する

```
echo "$time $opid $(~/DB_SAMPLE/
SHELL/temperture)" |
ssh pi@localhost "cat > ~/DB_SAMPLE/
DATA/INPUT/DATA.$opid.$time.$$"
EOS
chmod +x SHELL/SEND_TEMPERTURE
```

このコマンドではヒア・ドキュメント機能を使用してスクリプトを，SHELL/SEND_TEMPERTUREに書き込んでいます．気温データの作成（**図3**の③）で使用したヒア・ドキュメントと異なる点は，キーワードEOSがシングル・クオートで囲まれている点です．本来ヒア・ドキュメント中ではシェル変数の展開が行われるのですが，このようにすることで展開が抑制されます．ヒア・ドキュメント中は処理のスクリプトになります．

001番の観測器からの送信（**図22**）で実行したコマンドと異なる点は3行目の部分です．opid=$1となっています．$1はこれはこのスクリプトSHELL/SEND_TEMPERTUREを起動する際に，第1引き数に渡された値を参照して，シェル変数opidに代入するという記述です．このように記述することでこのスクリプト，SHELL/SEND_TEMPERTUREは，さまざまな観測地点のデータを送信するために使用できるようになります．

▶ステップ2：毎時0分に起動する設定を行う

cronデーモンに毎時0分にこのスクリプトを実行するように設定します．

ラズベリー・パイでは設定ファイルを編集するためのテキスト・エディタを選択するために，select-editorというコマンドを用意しています．このコマンドを実行すると次のような出力があるので，好きなエディタを選びましょう．ここでは各エディタの使い方には立ち入りません注2．

続いて，cornデーモンの設定を行うためにcrontabコマンドを実行します．

```
crontab -e
```

テキスト・エディタが起動し，cronデーモンの設定ファイルを編集することができます．ここで以下の1行を追加します．

```
0 *    *    *    *    ~/DB_SAMPLE/
SHELL/SEND_TEMPERTURE 001
```

このように記述することで毎時0分にSHELL/SEND_TEMPERTUREスクリプトが起動され，計測された気温が観測点番号001番の観測器のデータとして，データベース・サーバに送信されます注3．

ここで登場したコマンドは，温度を計測するコマンドを除いて全てラズベリー・パイ標準のコマンドです．UNIXにはこういった基本的な機能はだいたいそろっているので，うまく活用しましょう．

機能3：データの更新

● 基本手順

計測データが次々とDATA/INPUTディレクトリに到着するようになりました．この計測データを用いて，気温の時系列データのファイルを更新します（**図25**）．

▶ステップ1：作業ディレクトリへの移動

更新処理は，**図25**の①のように，処理対象のファイルを作業用ディレクトリに移動させるところから始まります．

```
pi@raspberrypi:~/DB_SAMPLE$ mv DATA/INPUT/DATA.*
DATA/INPUT.WORK
pi@raspberrypi:~/DB_SAMPLE$ cat DATA/TABLE/気温
DATA/INPUT.WORK/DATA.*
20160823T060000 001 18
20160823T070000 001 19
20160823T080000 001 21
20160823T060000 002 24
20160823T070000 002 26
20160823T080000 002 27
20160823T060000 003 22
20160823T070000 003 23
20160823T080000 003 24
20160823T060000 004 21
20160823T070000 004 23
20160823T080000 004 24
20160720T073630 001 23
20160720T073648 001 23
20160801T113524 001 23
pi@raspberrypi:~/DB_SAMPLE$ cat DATA/TABLE/気温
DATA/INPUT.WORK/DATA.*  |
> sort -k2,2 -k1,1  > DATA/INPUT.WORK/TABLE.NEW
pi@raspberrypi:~/DB_SAMPLE$ mv DATA/INPUT.WORK/
TABLE.NEW DATA/TABLE/気温
pi@raspberrypi:~/DB_SAMPLE$ mv DATA/INPUT.WORK/*
DATA/INPUT.DONE/
```

①作業ディレクトリへ移動
②データの結合
③ソート
④古いデータの上書き

図25　データの更新操作

注2：vimエディタについては，vimtutorというコマンドを実行することで使い方を学ぶことができる．
注3：この記述の意味は，man 5 crontabというコマンドを実行して表示されるマニュアルに詳しく記述されている．

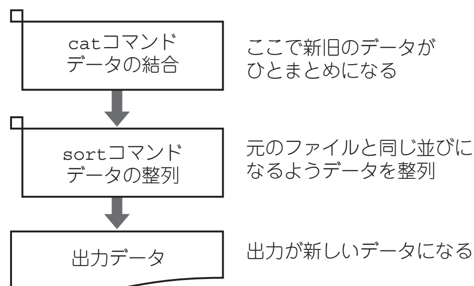


図26　更新データ作成処理

```
pi@raspberrypi:~/DB_SAMPLE$ cat   <<'EOS' > SHELL/UPDATE
> dir=~/DB_SAMPLE
> mv $dir/DATA/INPUT/DATA.* $dir/DATA/INPUT.WORK
> cat $dir/DATA/TABLE/気温 $dir/DATA/INPUT.WORK/DATA.*  |
> sort -k2,2 -k1,1     > $dir/DATA/INPUT.WORK/TABLE.NEW
> mv $dir/DATA/INPUT.WORK/TABLE.NEW $dir/DATA/TABLE/気温
> mv $dir/DATA/INPUT.WORK/* $dir/DATA/INPUT.DONE/
> EOS
pi@raspberrypi:~/DB_SAMPLE$ chmod +x SHELL/UPDATE
pi@raspberrypi:~/DB_SAMPLE$ chmod +x SHELL/SEND_TEMPERTURE
pi@raspberrypi:~/DB_SAMPLE$ crontab -e

②-2エディタが起動する．以下の通りに入力する
 10 *    *   *   *    ~/DB_SAMPLE/SHELL/UPDATE
```


図27　毎時10分に更新処理を行う

```
mv DATA/INPUT/DATA.* DATA/INPUT.
WORK
```

これは，どのファイルまで処理が終了したのかを明確にするためです．このようにすることで，処理開始時点で到着していたデータが全て作業領域に格納されます．

▶ステップ2:到着したデータと既にあるデータを結合

到着したデータと既にあるデータを結合します．

```
cat DATA/TABLE/気温 DATA/INPUT.
WORK/DATA.*
```

catコマンドは本来複数のファイルの内容を連結(conCATenate)するためのコマンドです．このようにすることで全ての内容が結合されます．

▶ステップ3：ソート

今回の気温ファイルは，観測点→観測日時の順に整列されていました．catコマンドで連結しただけでは，並び順がバラバラです．sortコマンドを使用して整列させます(**図26**)．

今回は説明のためにcatとsortの2段階で処理を記述します(**図25**の③)．

```
cat DATA/TABLE/気温 DATA/INPUT.
WORK/DATA.* |
sort -k2,2 -k1,1  > DATA/INPUT.
WORK/TABLE.NEW
```

これはsortのオプションを駆使することで1段階で記述することも可能です．

▶ステップ4：古いデータの上書き

これで新しい気温ファイルの内容がDATA/INPUT.WORK/TABLE.NEWに格納されました．後はこれを古い気温ファイルに上書きして，処理が終了したデータを処理完了ディレクトリに移動して完了です(**図25**の④)．

```
mv DATA/INPUT.WORK/TABLE.NEW DATA/
TABLE/気温
mv DATA/INPUT.WORK/* DATA/INPUT.
DONE/
```

● 毎時10分に更新処理を行う

更新処理は新しいデータが到着したら行います．またcronを利用して，各時刻の10分に更新処理を行うようにしてみます(**図27**)．

▶ステップ1：更新処理のスクリプトを作成

まず，更新処理を行うスクリプトを作成します(**図27**の①)．

```
cat  <<'EOS'    > SHELL/UPDATE
dir=~/DB_SAMPLE
mv $dir/DATA/INPUT/DATA.* $dir/
 DATA/INPUT.WORK
cat $dir/DATA/TABLE/気温 $dir/DATA/
 INPUT.WORK/DATA.*|
sort -k2,2 -k1,1  > $dir/DATA/
 INPUT.WORK/TABLE.NEW
mv $dir/DATA/INPUT.WORK/TABLE.NEW
 $dir/DATA/TABLE/気温
mv $dir/DATA/INPUT.WORK/* $dir/
 DATA/INPUT.DONE/
EOS
chmod +x SHELL/UPDATE
```

▶ステップ2：毎時10分に起動させる

続いて，crontab -eコマンドを実行して，以下の1行を追記して，このスクリプトを毎時10分に起動するように設定します．

```
10 *   *   *   *
~/DB_SAMPLE/SHELL/UPDATE
```

これで完成です．次々と到着するデータがDATA/TABLE/気温ファイルに格納されるようになりました．

なかむら・かずたか

サーバもクライアントも組み合わせてサクッ!

得意技②文字列処理&サーバ機能…インターネットでI/O

中村 和敬

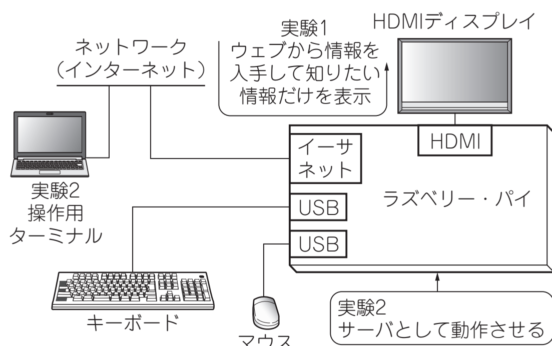


図1 インターネット・アクセスの実験環境

ラズベリー・パイには標準でイーサネット・インターフェースが装備されています．また，Wi-Fiはラズベリー・パイ3では標準で搭載されていますし，ラズベリー・パイ2でもUSBインターフェースを使って拡張できます．ラズベリー・パイでは標準でLinuxが動作しますので，インターネットを利用するためのソフトウェアは初めからそろっています．後はシェルを通じてコマンドを実行するだけでインターネットを利用する処理が可能です．

ここでは，シェルを通じてインターネットを利用するためのさまざまな方法を説明します．

実験環境を図1に示します．ラズベリー・パイはRaspbianを使っていて，DHCPでアドレスを取得できる環境であるならば，特段の設定の必要なくインターネットに接続できますので，ここではラズベリー・パイの設定に関する説明は省略します．

やること1：ウェブ・サーバからデータを取り出す

● 手順

シェルからサーバを使ってみます．試しに新潟県の人口を調べてみましょう．

▶ステップ1：知りたい情報のありかを確認

例えば，ウェブ・ブラウザで以下のURLを開いて，Wikipediaの新潟県のページを見てみます．



図2 データを取り出す簡易実験のターゲット…Wikipediaの新潟県のページ

```
https://ja.wikipedia.org/wiki/新潟県
```

すると，図2のようなページが開きます．総人口が2,290,569人と記載されていることが確認できます．

このような作業をシェルを通じて行ってみます(図3)．

▶ステップ2：ウェブ・ページの取得

ウェブ・ページなどをシェルから取得するには，`curl`コマンドを使用します．`curl`コマンドはブラウザなどで日常的に使用するプロトコルのほとんどに対応しており，ブラウザで済ませていた仕事をシェルで自動化したいというときには非常に便利です．ウェブ・ページをシェルから参照するには，図3の①のコマンドを実行します．

```
curl -s -L https://ja.wikipedia.org/wiki/新潟県
```

実行すると大量のテキスト・データがターミナルに出力されます．これは，Wikipediaの新潟県のページのHTMLソースコードです．ウェブ・ページをブラウザから見る場合，まずブラウザがURIにひも付く

```
pi@raspberrypi:~$ curl -s -L https://ja.wikipedia.org/wiki/新潟県  ←①ウェブ・ページの取得
<!DOCTYPE html>
<html lang="ja" dir="ltr" class="client-nojs">
<head>
<meta charset="UTF-8"/>
<title>新潟県 - Wikipedia</title>
<script>document.documentElement.className = document.documentElement.className.replace( /(^|\s)client-nojs(\
s|$)/, "$1client-js$2" );</script>^C

…中略…

pi@raspberrypi:~$ curl -s -L https://ja.wikipedia.org/wiki/新潟県|
> grep 総人口                                                      ←②探索範囲の絞り込み
<th scope="row" style="text-align:left; white-space:nowrap; background-color:#f0f0f0; text-align:center;
font-weight:normal;"><a href="/wiki/%E9%83%BD%E9%81%93%E5%BA%9C%E7%9C%8C%E3%81%AE%E4%BA%BA%E5%8F%A3%E4%B8%80%E8
%A6%A7" title="都道府県の人口一覧">総人口</a></th>
pi@raspberrypi:~$ curl -s -L https://ja.wikipedia.org/wiki/新潟県|
> grep -A1 総人口                                                  ←③目的の情報の範囲選択
<th scope="row" style="text-align:left; white-space:nowrap; background-color:#f0f0f0; text-align:center;
font-weight:normal;"><a href="/wiki/%E9%83%BD%E9%81%93%E5%BA%9C%E7%9C%8C%E3%81%AE%E4%BA%BA%E5%8F%A3%E4%B8%80%E8
%A6%A7" title="都道府県の人口一覧">総人口</a></th>
<td class="" style="" itemprop=""><b>2,290,569</b>人<br />
pi@raspberrypi:~$ curl -s -L https://ja.wikipedia.org/wiki/新潟県|
> grep -A1 総人口              |
> tail -1                      |                                   ←④目的の情報の切り出し
> tr '><' '\n'                 |
> grep -E '^[0-9,]+$'
2,290,569
```

図3　ウェブ・ページから知りたい情報を切り出す

HTMLデータを取得し，これを解釈して画面に表示します．出力されたテキスト・データは，ブラウザにより解釈される前のデータです．ブラウザで見るときのようなきれいな見た目ではありませんが，テキスト・データですので何が書いてあるのか読むことができますし，標準のUNIXコマンドを使って操作をすることができます．

▶ステップ3：探索範囲の絞り込み

HTMLソースコードから総人口が記述されている部分を探してみます(**図3**の②)．

```
curl -s -L https://ja.wikipedia.org/
wiki/新潟県    |
grep 総人口
```

curlコマンドからの出力をgrepコマンドを使用して検索して出力します．grepコマンドは入力データを指定されたパターンについて行単位で検索するコマンドです．

grepコマンドの名前は，UNIXの古いテキスト・エディタedの「ファイル全体から/正規表現に一致する行を/表示する」コマンドg/re/pに由来します．grepコマンドの探索するパターンは，「正規表現」と呼ばれる記法で記述されます．正規表現を学ぶことで複雑な検索を行うことができます．

図3の②のコマンドを入力すると，次のような出力が得られます．

```
<th scope="row" style="text-
align:left; white-space:nowrap;

…中略…

title="都道府県の人口一覧">総人口</a></
th>
```

総人口の数値がありません．実は総人口の数値はこの行の次の行に含まれます．

▶ステップ4：目的の情報の切り出し

発見したパターンが含まれる行の後，何行かを出力する-Aオプションを付けて，再度実行してみます(**図3**の③)．

```
curl -s -L https://ja.wikipedia.org/
wiki/新潟県    |
grep -A1 総人口
```

-A1を指定しているので，パターンが含まれる行の次の行が表示されます．

```
<th scope="row" style="text-
align:left; white-space:nowrap;

…中略…

<b>2,290,569</b>人<br />
```

人数が表示されています．後はここから人口の部分だけを切り出します(**図3**の④)．

```
curl -s -L https://ja.wikipedia.
org/wiki/新潟県 |
grep -A1 総人口 |
tail -1         |
tr '><' '\n'    |
```

```
pi@raspberrypi:~$ cat  <<'EOS' > 総人口検索スクリプト
> name=$1
> curl -s -L https://ja.wikipedia.org/wiki/$name |
> grep -A1 '>総人口<'           |
> tail -1                      |
> tr '><' '\n'                 |
> grep -E '^[0-9,]+$'
> EOS                                       ②スクリプトの実行
pi@raspberrypi:~$ chmod +x 総人口検索スクリプト
                                            ①スクリプトの作成
```

図4　総人口検索スクリプトの作成

```
grep -E '^[0-9,]+$'
```

すると，2,290,569と人口だけが出力されました．このコマンドは，先ほどの出力を3行目で人口が含まれている行のみを取り出しています．その後，4行目のtrコマンドで>と<を改行に変換しています．最後の5行目は少々複雑な拡張正規表現を使用して，人口の行のみを取得しています．

● スクリプトの作成

他の県についてもほとんど同じコマンドで検索が可能です．そこで，総人口検索スクリプトを作成してみます(**図4**)．スクリプトの処理の流れを**図5**に示します．

```
cat <<'EOS'      > 総人口検索スクリプト
name=$1
curl  -s  -L  https://ja.wikipedia.
org/wiki/$name|
grep -A1 '>総人口<'          |
tail -1                       |
tr '><' '\n'                  |
grep -E '^[0-9,]+$'
EOS
chmod +x 総人口検索スクリプト
```

各都道府県の総人口を取得するには，**図4**の②のようにします．

```
./総人口検索スクリプト 北海道
```

シェルを使用することで簡単にウェブから必要なデータを取得できます．こういった技術をウェブ・スクレイピングといいます．シェルでは既に用意されているコマンドで多くのことが可能です．より詳細な分析が必要な場合には，grepをもっと使いこなしたり，HTMLをきちんと解釈してデータを取り出すことのできる，xmllintコマンドなどを使用するのがよいでしょう．

● 注意点…サーバ側の仕様が変更になると処理を見直さないといけない

近年はウェブを通じてさまざまなサービスが提供され，サービスを利用するためのHTTPを利用したAPI，WebAPIが公開されています．そういったさまざまなウェブ・サービスも，ここで紹介した方法と同様の方法で利用できます．

注意点として，ウェブ・サーバのデータの形式やWebAPIは変更される場合があります．例えば，上述の例ではcurlコマンドを呼び出す際に-Lオプションを使用していますが，これは以前は必要ありませんでした．このように，サーバ側の仕様の変更のたびに処理を見直す必要があることに留意しましょう．

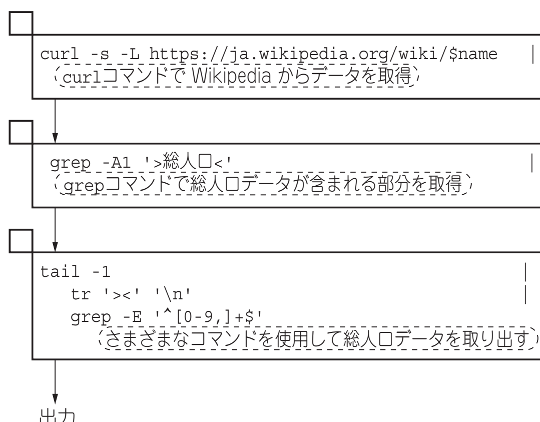


図5　総人口検索スクリプト処理

やること2：超簡易日付けサーバの構築

シェルでサーバを作成して，立ち上げてみます[注1]．

● 準備

サーバを作成するためにsocatというコマンドを使用します(**図6**)．socatは「クライアントの接続の待ち受け」～「接続ごとに応対プロセスの立ち上げ」の処理を行ってくれます．

socatコマンドは標準ではインストールされていないので，次のコマンドを実行してインストールします．

```
sudo apt-get install socat
```

クライアントの接続の待ち受け機能については，socatが行ってくれるので，接続ごとに応対するプログラムがあれば，サーバを立ち上げることができます．

注1：サーバの立ち上げでは，ネットワーク・セキュリティの観点から非常に危険な操作が含まれている．試してみる際には，必ずファイアウォールで保護されているなどの，安全なネットワークを利用する．

①サーバ側の操作

```
pi@raspberrypi:~$ socat tcp-listen:10000,reuseaddr,fork exec:date
```

このコマンドを起動すると<Ctrl-C>を押すまで終了しない．このため，サーバの操作は，別のターミナルから行う必要がある．サーバの終了は<Ctrl-C>

②端末側の操作

```
pi@raspberrypi:~$ socat tcp:localhost:10000 stdio
Mon  1 Aug 12:05:57 UTC 2016
pi@raspberrypi:~$
```

図7　接続があると日付を返すサーバを立ち上げる

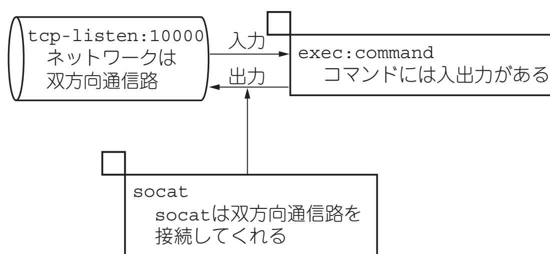


図6　socatコマンドの処理

● 手順

▶ステップ1：サーバの立ち上げ

接続があると日付を返すサーバを立ち上げます(**図7**の①)．

```
socat tcp-listen:10000,reuseaddr,fork
exec:date
```

これは，TCPポート10000番でクライアントの接続を待ち受け，接続があったら，接続応対用のプログラムとしてdateコマンドを実行するというコマンドです(**図8**)．

これだけでサーバが立ち上がりました．何もしなければコマンドは動き続け，サーバはクライアントからの接続を待ち続けます．サーバを終了させるには，<Ctrl-C>を入力します．

▶ステップ2：サーバへの接続

立ち上げたサーバを利用してみます(**図7**の②)．

```
socat tcp:localhost:10000 stdio
```

これは，自身のTCPポート10000番で待ち受けるサーバに接続し，サーバからの入出力を標準入出力に接続するというコマンドです．このコマンドを実行すると，ターミナルにdateコマンドを実行したときと同じ出力がされて，コマンドが終了します．

```
Mon  1 Aug 12:05:57 UTC 2016
```

● ファイアウォールの設定

上述のコマンドはサーバを立ち上げたコンピュータと同じコンピュータでクライアントを立ち上げサーバに接続しましたが，他のコンピュータからサーバに接続することももちろん可能です．別のコンピュータから上述のコマンドのlocalhostをサーバのIPアドレス，またはホスト名に変えて実行します．同じ出力を確認できます．別のコンピュータからサーバに接続する際には，ファイアウォールの設定などを確認して，TCPの10000番ポートを通すようになっていることを確認してください．

● 簡易socatコマンド・サーバの次のステップ

さて，これだけでサーバができてしまいました．接続を応対するプログラムを変えれば，いろいろなサーバを立ち上げることができます．

例えば以下のコマンドは接続するとBashというシェルが接続を応対するので，シェル・コマンドが実行できてしまいます．

```
socat tcp-listen:10000,reuseaddr,fork
exec:bash
```

このコマンドはネットワーク・セキュリティの観点から非常に危険です．試してみる際には，必ずファイアウォールで保護されているなどの安全なネットワークを利用してください．

このようにシェルでも簡単にサーバを開発できてしまいます．本格的なサーバについても，HTTPなどのテキスト・ベースのプロトコルであれば，awkコマンドなどを用いて作成することができます．

socatはテスト用のコマンドとしての性格が強く，サーバとして本格的に運用するために必要な機能に欠けています．しかし，socatを利用して作成した接続を応対するプログラムは，xinetdなど，より本格的な汎用サーバから起動することができます．

なかむら・かずたか

第2特集 Appendix 3

基本思想や動作の理解で差がつく！

データ主義！良いシェル・プログラミングの勘どころ

中村 和敬

```
pi@raspberrypi:~$ cat  <<EOF   > file
> 1
> 2
> 3
> 8
> 9
> 0
> EOF
```

図1　1行ごとに1～9の数値データが格納されているファイルの作成

リスト1　ファイルに組まれている数を合計して標準出力に出力する処理

```
sum=0
for i in $(cat file) ; do
    sum=$(($sum + $i))
done
echo $sum
```

(a) for文を使用した場合

```
sum=0
cat file    |
while read i ; do
    sum=$(($sum + $i))
done
echo $sum
```

(b) while文を使用した場合

シェルはUNIXのユーザ・インターフェースです．本格的なプログラム言語なので，UNIXの機能を使って本格的なシステムを作ることができます．しかしシェルを用いて，分かりやすく，高速なプログラムを記述するには，C言語やJavaなどの他のプログラム言語とは異なる考え方が必要となります．本稿ではシェルでプログラミングを行う際の考え方を説明します．

シェル・プログラミングを書く前に知っておいた方がよい注意点

シェルはC言語やJavaなどのプログラム言語と同様に，変数を扱うことができます．しかし，通常のプログラム言語と同じような考え方でシェル変数を使うと，さまざまな問題が発生します．

● その1：while文の変数スコープが変則的

第1が変数のスコープの問題です．今，ファイルfileに，1行ごとに何らかの数値データが格納されているとしましょう．簡単のため1～9の数が入っているとします(**図1**)．

さて，このファイルに組まれている数を合計し，標準出力[注1]に出力する処理を書きたいとしましょう．

このとき**リスト1**に示す2通りの処理が考えられます．**リスト1(a)**は正しい結果(45)が出力されますが，**リスト1(b)**は誤った結果(0)が出力されてしまいます．これはシェルのサブシェルという挙動に由来します．

for文を使用した場合は通常のプログラム言語と同じように，シェル変数sumに次々と数値が加算されていきます．

while文を使用した場合には，少し挙動が異なります(**図2**)．まず，元のシェルのプロセス中でシェル変数sumに 0 が代入されています．続いて，seqコ

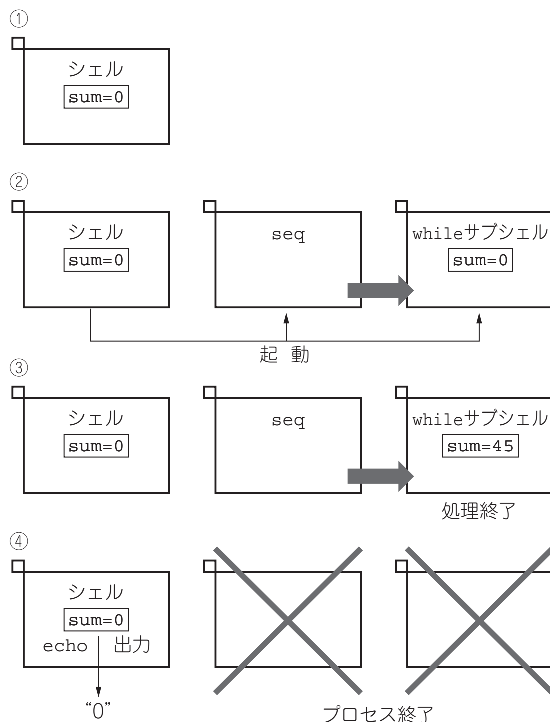


図2　while文によるサブシェル起動の際のプロセスとシェル変数の遷移

注1：標準出力とは，UNIXが用意するプログラムの標準的な出力のことで，通常シェルの開いているターミナル．

マンドのプロセスと，while文を処理するシェルのプロセスが起動されます．この，while文を処理するシェルのプロセスをサブシェルと呼びます．

サブシェルには元のシェルのシェル変数sumがコピーされます．この時点でsumは0です．処理が進むにつれて，while文のシェルのプロセスのシェル変数sumに数値が加算されていき，最後は45になります．最後まで加算をすると，while文のシェルのプロセスは終了します．この時点で，45まで加算されたシェル変数sumはwhile文のシェルのプロセスのプロセスと共に消えてしまいます．そして，その後でシェル変数sumをechoしても，元々のシェルのプロセスのシェル変数sumの値だった0が出力されるというわけです．

● その2：シェル変数を使うと処理速度がすごく遅くなる

第2が処理速度の問題です．シェルはプログラム言語ではありますが，それ自体の処理速度は速くはありません．実際には後で述べるように，シェルを使用して高速な処理を記述することができます．それにはシェル変数を用いたプログラムをしてはいけません．

このように，シェルをC言語やJavaなどのプログラム言語と同様の感覚でプログラムを行うと，いろいろな落とし穴にはまってしまいます．

良いシェル・プログラムを書く基本思想…データの流れをプログラムする

シェルで良いプログラムを書くためには，データの流れをプログラムする，という考え方が重要です(**図3**)．シェル自体で処理をプログラムするのではありません．

シェルでプログラムをする際には，シェルから起動されるコマンドの入出力をファイルやパイプで接続して，データの流れをプログラムします．接続にはできるだけパイプを利用し，実際の処理はシェルから起動されたコマンドに行わせます．そのようにプログラムすることで，分かりやすく，高速なプログラムを記述することができます．

シェルでプログラムする際には，メインの処理にシェル変数を使用してはいけません．あくまで全体の処理の流れを設定するために使うにとどめるのが，良い使い方であるといえます．

リスト2　ファイルに組まれている数を合計して標準出力に出力する処理（awkによる記述）

```
cat file             |
awk -v sum=0 ' { sum+=$0 }
    END { print sum }'
```

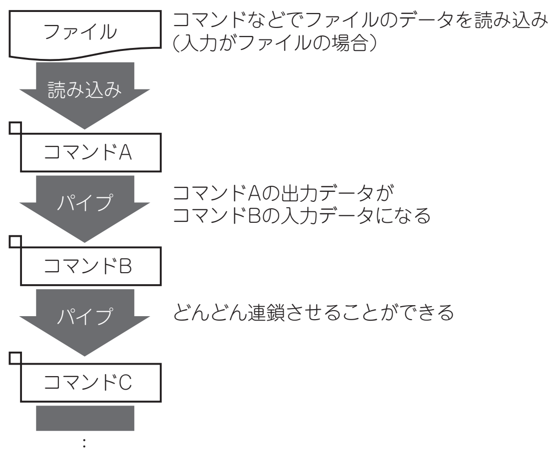


図3　データの流れをプログラムする

良いシェル・プログラムのご利益

● その1：コマンドを使って処理性能を劇的に上げられる

先ほどの合計を求める例を考えてみましょう．awkコマンドを使って**リスト2**のように書き換えます．

リスト1ではfor文やwhile文を使って合計を計算する処理を記述していました．**リスト2**ではawkコマンドを使用して合計を計算しています．また，処理の結果はシェル変数に代入した内容をechoコマンドで出力するのではなく，直接awkコマンドから出力しています．この場合，先ほどのようなシェル変数のスコープの問題は発生しません．また，速度面でも大きな改善になります．差が分かりやすいように，1～100,000の数値を格納したファイルfile.bigを作成して，シェルのfor文を使用する場合と，awkコマンドにより処理を行う場合の両方でtimeコマンド[注2]を使用して処理時間を計測して比べてみましょう(**図4**)．

```
pi@raspberrypi:~ $ seq -f'%.0f' 1 100000 > file.big
pi@raspberrypi:~ $ time bash -c 'sum=0;for i in
$(cat file.big) ; do sum=$(($sum + $i)) ; done;echo
$sum'
5000050000

real    0m7.892s
user    0m7.820s    } for文を使用した場合
sys     0m0.070s
pi@raspberrypi:~ $ time cat file.big | awk -v sum=0
'{ sum+=$0 }END{ printf("%.0f\n", sum); }'
5000050000

real    0m0.178s
user    0m0.170s    } awkを使用した場合
sys     0m0.020s
```

図4　awkを使用すると処理時間を短縮できる

リスト3　ファイル中の偶数だけを合計する処理

```
sum=0
for i in $(cat file) ; do
    if [ $(($i % 2)) -eq 0 ] ; then
        sum=$(($sum + $i))
    fi
done
echo $sum
```

(a) シェル変数を使用してプログラム

```
cat file        |
awk '$0 % 2 == 0'  |
awk -v sum=0 ' { sum+=$0 }
    END { print sum }'
```

(b) データの流れをプログラム

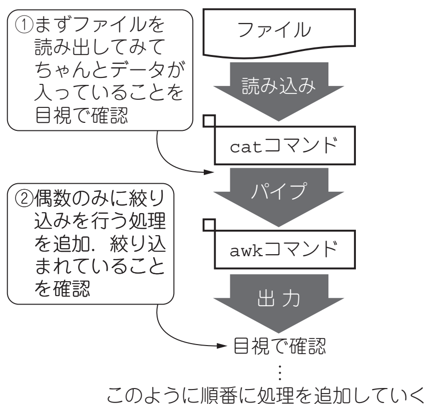


図5　データの流れを作り出し段階を追って処理を行う

awkを使用した場合, 処理時間は約1/44になりました. これは処理をシェルではなく, コマンドで行った結果です. コマンドの方が, そのコマンドの機能に最適化されているので, より高速に処理を行えます.

● その2：複雑な処理プログラムも作りやすい

データの流れをプログラミングするという考え方は, 複雑な処理を記述する際にも力を発揮します. 例えばファイル中の偶数だけを合計する処理を記述するとしましょう. シェル変数を使用した記述は**リスト3(a)**のようになります. データの流れをプログラムするという観点で記述すると**リスト3(b)**のようになります. シェル変数を使用した場合, for文の中のネストが深くなる形で記述を行います.

一方データの流れをプログラムするという観点で記述を行った場合, 1行目でファイルを読み出した後, 2行目で偶数のみを取り出します. そして3行目で合計を求めています.

この処理の開発方法は, for文による方法よりははるかに理解しやすく, 開発しやすいものです.

まず1行目でファイルを読み出します.

```
cat file
```

続いて合計すべきデータの絞り込みを行います. 1行目のデータの処理に, 偶数のデータのみを取り出す処理をパイプで接続します.

```
cat file            |
awk '$0 % 2 == 0'
```

たった今自分の記述した処理が正しいか, 出力を直接見て確認できます. そして最後に合計を行う処理を追加して完成します.

このようにシェルでプログラムを行う際は, データの流れを作り出し, 段階を追って処理を行うという考え方で処理を記述していきます(**図5**).

こういった形で処理を記述できるということは, 複雑なデータ処理を記述する際に威力を発揮します. 増改築を繰り返して解読不能になったSQLプログラムなどを, 段階を追って処理を行う, 分かりやすいプログラムに書き換えることができます.

● その3：処理が順番ずつ書いてあるので読みやすい

シェル・スクリプトは適切なコマンドを使用することで, 可読性が非常に高くなります. 例えば, 先ほどの偶数の合計を求める処理を**リスト4**のように記述すると,

①ファイルfileを読み出し
②偶数の行を絞り込んで
③全ての行を足す

という処理がより明確になりました. getevenlineやsumallといったコマンドはUNIX標準ではありません. しかし, 必要に応じて注意深くコマンドを開発し, 使用することで, 記述した処理の内容がより明確になり, 可読性が向上します(**図6**).

リスト4　ファイル中の偶数だけを合計する処理の可読性を高める

```
cat file    |
getevenline |
sumall
```

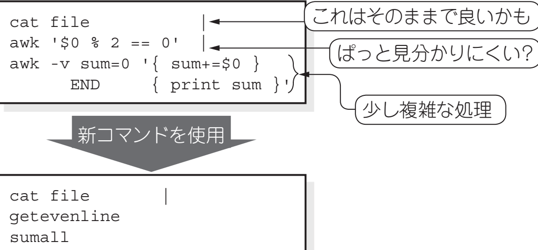


図6　処理をまとめてコマンドを作る

注2：timeコマンドは実行したコマンドの処理時間を表示する. リソースの使用量も表示できるよう拡張されたものもある. シェル・スクリプトのチューニングに活躍する.

なかむら・かずたか

◆参考・引用＊文献◆

(1) 中島 雅弘, 富永 浩之, 國信 真吾, 花川 直己；AWK実践入門, 技術評論社, 2015年.

自分専用の機能を高速で使いやすく!

レベルアップ! オリジナル・コマンドを作る

中村 和敬

UNIXの提供するコマンド群では，目的のシステムを構築できなかったり，できたとしても複雑になってしまったり，保守が難しくなったりしてしまう場合があります．そのような場合には，まず他に実現可能なコマンドが存在しないか確認し，その上でないようならば新しいコマンドを開発します．

コマンドの開発には，好きなプログラム言語を用いることができます．C言語はもちろん，RubyやPythonなどのスクリプト言語，もちろんシェル・スクリプトでも構いません．

しかし，必要ならすぐにコマンドを作ればよいというものではありません．注意深く機能を絞り込んでコマンドを実装することで，そのコマンドはUNIXの一部として環境に組み込まれ，後々まで役立つ便利な部品の一つになるでしょう．

本稿では，まず新しいコマンドを開発する際の考え方を紹介します．続いて，シェル・スクリプトを用いて新しいコマンドを開発する過程を説明します．最後に，名前付きパイプという仕組みを使用したシェル・スクリプトの高速化手法も紹介します．

どんなときに新しいコマンドを作るか

コマンドを開発した方がよい場合を考えてみましょう．

● 場面1：汎用的な処理

第3章では，図1のような，行指向スペース区切りの形式のデータを取り扱いました．

```
20160823T060000 001 18
20160823T070000 001 19
20160823T080000 001 21
20160823T060000 002 24
20160823T070000 002 26
20160823T080000 002 27
20160823T060000 003 22
20160823T070000 003 23
20160823T080000 003 24
20160823T060000 004 21
20160823T070000 004 23
20160823T080000 004 24
```

図1　行指向スペース区切りの形式のデータ

ここで，それぞれの観測地点の6時から7時の間の平均気温を計算するには，どうしたらよいでしょうか．例えばリスト1のようなコマンドで計算できます．

2行目のawkスクリプトで絞り込みを行い，3行目のawkスクリプトで計算しています(図2)．

3行目では変数sumに気温の合計，countにレコード数を代入し，異なる地点のデータが到着したとき，あるいは全てのデータを読み込み終わったときに，平均値を計算して出力します．

2行目のawkコマンドの第2引き数に入力ファイルを渡すことで1行目のcatはなくせますが，シェル・スクリプトはデータが左から右，上から下に流れるということを考えると，このような書き方の方が読みやすくなり，デバッグの際などにも有効です．

● 場面2：既存のコマンドで簡潔な記述ができない

こういった計算が一度で済めばよいのですが，平均を求めるような処理は一般的で，他のさまざま場面で必要とされます．すると，このスクリプトを毎回それぞれのデータに合わせて書き換えなければなりません．

このawkスクリプトは決して難しくはないのですが，一目で処理の内容が分かるわけでもありません．また，このような書き方以外にもいろいろな書き方が可能です．そういったスクリプトをいちいち書き換えていくと，バグの温床となるだけでなく，保守性も低下してしまいます．この場合は新しくコマンドを開発した方がよいでしょう．

リスト1　それぞれの観測地点の6時から7時の間の平均気温を計算する

```
cat DATA/TABLE/気温                                      |
awk '"060000"<=substr($1, 10, 6) && substr($1, 10,
                              6) <= "075959"'        |
awk '   NR==1   { key=$2; count=0; sum=0; }
        key==$2 { count+=1; sum+=$3 }
        key!=$2 { print key, sum / count
                  key=$2; count=1; sum=$3; }
        END { print key, sum / count }'
```

パターン	アクション	
NR==1	{ key=$2; count=0; sum=0; }	1行目のとき、これ以降の処理に備えてkey, count, sumを初期化
key==$2	{ count+=1; sum+=$3 }	keyが同じレコードのとき, countをインクリメントし, sumに平均を求める対象の第3フィールドを加算
key!=$2	{ print key, sum / count key=$2; count=1; sum=$3; }	keyが変わったとき, keyと平均を出力して, これ以降の処理に備えてkey, count, sumを初期化
END	{ print key, sum / count }	全ての行を処理し終わったら, keyと平均を出力

図2　平均計算awkスクリプトの処理

● 場面3：速度が出ない

また, 他にもスクリプトで記述していてはどうしても速度が出ないという場合にもコマンド化することがあります. しかし, 移植性が低下しコマンドの保守コストもかかるので注意した方がよいでしょう.

機能の決定

新しいコマンドを開発する際に, どのような機能のコマンドを開発すればよいのか, もう一度考えてみます.

● 必要な機能を整理する

今回必要な機能は, 以下の通りです.

- レコードを特定の条件で絞り込み
- 複数のレコードを指定されたフィールド(以下キー)の値が同じもので集約
- 指定されたフィールドの値の平均を求める

今回のデータは, あらかじめ観測点→観測日時の順に整列されたデータなので,「複数のレコードを指定されたフィールドの値が同じもので集約」の部分は, 初めから同じキーのレコードが並ぶように整列されていました. また, 平均はデータの合計をデータの数で割ったものです.

● 機能を細かく分解する

ここから, 先ほどの機能は, 以下の機能に分解できます.

①レコードをキーの値により整列する
②レコードを特定の条件で絞り込む
③それぞれのキーの値について, 同じキーの値を持つレコードの個数を数える
④それぞれのキーの値について, 同じキーの値を持つレコードの, 特定のフィールドの合計を計算する
⑤それぞれのレコードについて, 特定のフィールドの間での除算を行う

①については, 既にデータを整列するためのsortコマンドがあります. また②の絞り込みについては, 平均を求める処理とは明らかに別の機能です. そこで①と②については今回作成するコマンドの機能からは除外しましょう.

③, ④, ⑤は, いずれも汎用的な処理です. データの個数を数えたり, 合計を求めたり, フィールドの値を用いての演算は他の部分でも使うことがありそうです. むしろその頻度は平均を求めることよりも多いでしょう.

● 分解した機能に合わせて記述する

そこで, **リスト1**のawkを用いた処理を, これらの機能が切り分けやすい形に書き換えてみます(**リスト2**).

リスト1と**リスト2**が同じ出力になることを確認できたでしょうか. **リスト2**の③と④の部分が**リスト1**とあまり変わりがありません.

そこで③と④の機能をそれぞれ別のコマンドとして作成して, その後③, ④, ⑤を組み合わせて平均を求める機能を作ります.

処理を一般化

● 同じキーの値を持つレコードの個数を数えるコマンドを作る

ここでは,「それぞれのキーの値について, 同じキーの値を持つレコードの個数を数える」コマンドを作ってみます.

リスト2の③のコマンドでは, キーは第2フィールド, 値は第3フィールドで固定になっています. これを一般化して, いろいろなデータに対して使用できるシェル・スクリプトのファイルを作成します(**リスト3**).

第1引き数に何フィールド目がキーであるか, 第2引き数に処理対象のファイルを指定して起動します. 第2引き数を省略すると標準入力から処理対象のデータを読み込みます.

試しに先ほどの処理をこのコマンドを使用して書き直し, 処理結果を比較してみます(**図3**).

1行目で今作成したコマンドを実行して, 結果をtmp-count.newというファイルに格納していま

リスト2　機能を切り分けやすい形に書き換える

```
cat DATA/TABLE/気温         |
awk '"060000"<=substr($1, 10, 6) &&
    substr($1, 10, 6) <= "075959"'> tmp-data

cat tmp-data                |
awk '   NR==1   { key=$2; count=0; }
    key==$2 { count+=1; }
    key!=$2 { print key, count
          key=$2; count=1; }
    END { print key, count }'  > tmp-count


cat tmp-data                |
awk '   NR==1   { key=$2; sum=0; }
    key==$2 { sum+=$3; }
    key!=$2 { print key, sum
          key=$2; sum=$3; }
    END { print key, sum }'    > tmp-sum

join -1 1 -2 1 tmp-count tmp-sum  |
awk '{ print $1, $3 / $2 }'
```

①，②整列されたデー タのレコードを特定の条件で絞り込む

③ それぞれのキーの値について，同じキーの値を持つレコードの個数を数える

④ それぞれのキーの値について，同じキーの値を持つレコードの，特定のフィールドの合計を計算する

⑤ それぞれのレコードについて，特定のフィールドの間での除算を行う

リスト3　同じキーの値を持つレコードの個数を数えるコマンドSHELL/count

```
cat <<"EOS" > SHELL/count
awk -v keyind=$1   '
    NR==1            { key=$(keyind); count=0; }
    key==$(keyind) { count+=1; }
    key!=$(keyind) { print key, count
                     key=$(keyind); count=1; }
    END              { print key, count }
'   ${2:-'-'}
EOS
chmod +x SHELL/count
```

リスト4　同じキーを持つフィールドの合計を計算するコマンドSHELL/sum

```
cat <<"EOS" > SHELL/sum
awk -v keyind=$1 -v valind=$2 '
    NR==1            { key=$(keyind); sum=0; }
    key==$(keyind) { sum+=$(valind); }
    key!=$(keyind) { print key, sum
                     key=$(keyind); sum=$(valind); }
    END              { print key, sum }
'   ${3:-'-'}
EOS
chmod +x SHELL/sum
```

```
pi@raspberrypi:~/DB_SAMPLE$ SHELL/count 2 tmp-data   > tmp-count.new
pi@raspberrypi:~/DB_SAMPLE$ diff tmp-count tmp-count.new
```

図3　SHELL/countの実行例

```
pi@raspberrypi:~/DB_SAMPLE$ SHELL/count 2 tmp-data   > tmp-count.new
pi@raspberrypi:~/DB_SAMPLE$ diff tmp-count tmp-count.new
```

図4　SHELL/sumの実行例

す．2行目は古い記述の出力，tmp-countと，新しいコマンドを使用した記述の出力，tmp-count.newを比較して差分を出力します．今回は差がないので，なにも出力されません．

● 同じキーを持つフィールドの合計を計算するコマンドを作る

同じように，**リスト2**の④の「それぞれのキーの値について，同じキーの値を持つレコードの，特定のフィールドの合計を計算する」コマンドも作ります．

これを一般化して作成したコマンドが**リスト4**です．第1引き数に何フィールド目がキーであるか，第2引き数に何フィールド目が合計する対象のフィールドか，第3引き数に処理対象のファイルを指定して起動します．第3引き数を省略すると標準入力から処理対象のデータを読み込みます．

同じようにこのコマンドを使用して書き直し，処理結果を比較してみます(**図4**)．

ここで開発したコマンドを使用した場合，**リスト2**の処理の記述は**リスト5**のようになります．比べてみると少し簡単になったことが分かります．

名前付きパイプを使って高速化

今開発したコマンドを使用して，平均を求めるコマンドを作ってみます．

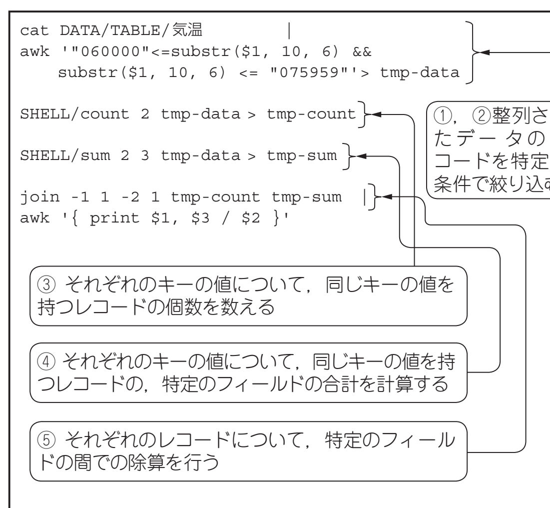
● 平均を求めるスクリプトを作成

平均を求める**リスト5**の処理を一般化してスクリプトを作成すると**リスト6**のようになります．

第1引き数に何フィールド目がキーであるか，第2引き数に何フィールド目が平均を求める対象のフィールドか，第3引き数に処理対象のファイルを指定して起動します．

スクリプトの1行目は一時ファイルの名前として利用する，tmpという名前のシェル変数を定義しています．シェル変数\$\$はプロセスの識別番号(プロセスID)に展開されるので，他のプロセスと一時ファイルの名前がかぶることはありません．

リスト5　一般化した機能によるコマンドを組み合わせて平均を計算する



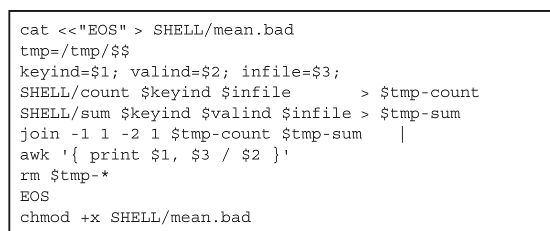
```
cat DATA/TABLE/気温          |
awk '"060000"<=substr($1, 10, 6) &&
    substr($1, 10, 6) <= "075959"'> tmp-data

SHELL/count 2 tmp-data > tmp-count

SHELL/sum 2 3 tmp-data > tmp-sum

join -1 1 -2 1 tmp-count tmp-sum  |
awk '{ print $1, $3 / $2 }'
```


リスト6　平均を求める処理を一般化して作成したコマンド SHELL/mean.bad



```
cat <<"EOS" > SHELL/mean.bad
tmp=/tmp/$$
keyind=$1; valind=$2; infile=$3;
SHELL/count $keyind $infile       > $tmp-count
SHELL/sum $keyind $valind $infile > $tmp-sum
join -1 1 -2 1 $tmp-count $tmp-sum    |
awk '{ print $1, $3 / $2 }'
rm $tmp-*
EOS
chmod +x SHELL/mean.bad
```

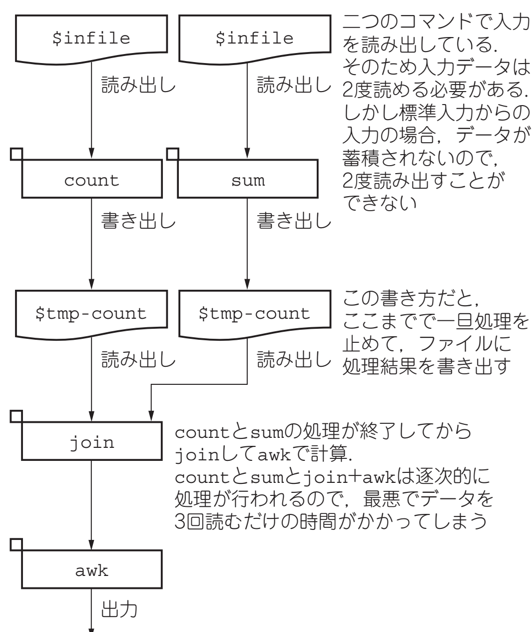


図5　ファイル読み出しの場合のデータの流れ

● ただ記述するだけではフィルタとして機能しない

リスト6には，**図5**のように，幾つかの問題があります．まず，このコマンドはフィルタとして機能しません．標準入力からデータを読み込むことができないのです．また，SHELL/countコマンドとSHELL/sumコマンドのそれぞれの処理の後で，一度ファイルに結果を書き出してから処理を行っています．そのため，非常に大きいデータを処理する際には，処理速度が悪化してしまいます．先ほどの標準入力からデータを読み込めないという問題は，一度データをファイルに書き出してから処理をすればよいのですが，やはり同じように性能が悪化してしまいます．

● 名前付きパイプを使う

フィルタとして利用できないというだけでも，コマンドとしてみれば大変使い勝手が悪くなってしまいます．なんとかして解決しなければなりません．

このようなときには，UNIXの名前付きパイプという機能を使用します．名前付きパイプはあたかもファイルのように扱えるパイプです．これを使用してスクリプトを書き換えると**リスト7**のようになります．

行末に「&」が付いている行は，バックグラウンド処理という，並列に処理を起動するための仕組みを利用するための記述です．バックグラウンド処理と名前付きパイプを活用することで，count，sum，join，awk全てのコマンドは並列に動作します（**図6**）．

最後に元の処理をこのコマンドを使って書き換えてみます（**リスト8**）．

だいぶ簡潔に書けるようになりました．これなら間違えることはなさそうです．

● パイプを詰まらせないように注意する

実際にコマンドを開発する際には，C言語などの高速な処理を記述できる言語を使うことが一般的です．例えば先ほどのSHELL/countコマンドやSHELL/sumコマンドは最終的にC言語で開発した方がよいでしょう．

しかし，SHELL/meanコマンドについては，内部で使用しているコマンドが十分に高速であれば，C言語などで開発したものと比べても遜色のない性能を出すこともしばしばです．

名前付きパイプを使用する際の注意点として，パイプが詰まってしまう場合があるということです．今回のような一度分岐して再度合流するような処理の場合，合流する出力の量が異なる場合などにパイプの詰まりが発生しやすくなります．

コラム　コマンドの設計思想

中村 和敬

コマンドは起動するたびに処理を行い，終了するプログラムです．UNIXにはさまざまなコマンドがありますが，以下のような設計思想に基づいて開発されています．

まず，コマンドはフィルタという体裁で作成されることが推奨されています．UNIXのプロセスは，標準入力，標準出力，標準エラー出力という標準的な入出力を持ちます．フィルタとは必要に応じて標準入力からデータを読み込み，処理をして，標準出力に書き出すようなプログラムのことです．

また，コマンドはそれぞれ限られた小さな機能を提供することが推奨されています．それぞれのコマンドは目的を達成するための部品として作られているからです．カーネルはパイプという機能を提供しており，この機能を用いることで，あるプロセスの標準出力への出力を，別のプロセスの標準入力に対する入力とすることができます．これによりユーザは必要な機能をもったコマンドを組み合わせて目的を達成することができます．

リスト7　名前付きパイプを使って書き換える

```
cat <<"EOS" > SHELL/mean
tmp=/tmp/$$
keyind=$1; valind=$2; infile=${3:-'-'};
mkfifo $tmp-infile.count $tmp-infile.sum $tmp-count
$tmp-sum
cat $infile    |
tee $tmp-infile.count  > $tmp-infile.sum          &
SHELL/count $keyind $tmp-infile.count     > $tmp-
                                            count   &
SHELL/sum $keyind $valind $tmp-infile.sum > $tmp-sum
                                                    &
join -1 1 -2 1 $tmp-count $tmp-sum    |
awk '{ print $1, $3 / $2 }'
rm $tmp-*
EOS
chmod +x SHELL/mean
```

リスト8　作成したコマンドを使って6時から7時の間の平均気温を計算する処理を書き換える

```
cat DATA/TABLE/気温                                  |
awk '"060000"<=substr($1, 10, 6) && substr($1, 10,
                              6) <= "075959"'      |
SHELL/mean 2 3
#+END_SRC
```

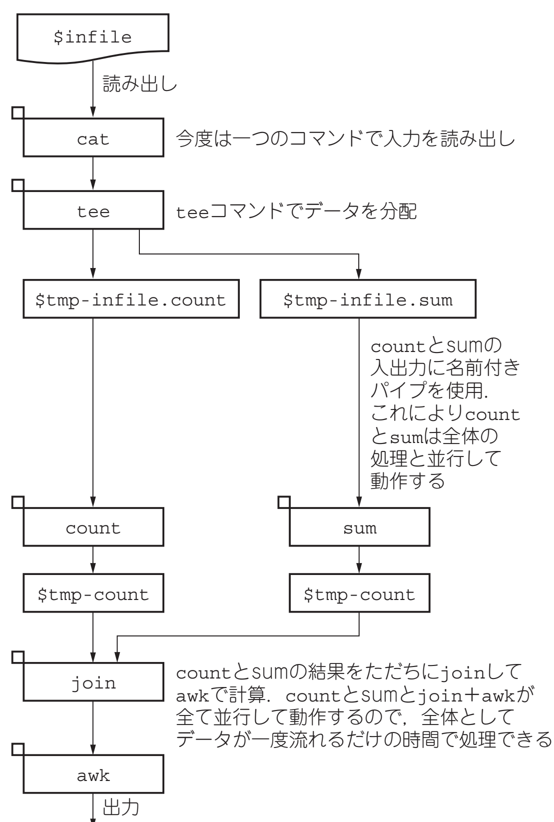


図6　名前付きパイプを使用した場合のデータの流れ

＊　＊　＊

今回は例としてコマンドを開発してみました．しかし，コマンドは本来軽々しく開発するものではありません．というのも，大量のコマンドがあっても人はそれらを覚えきれないからです．覚えていないコマンドは使われず，せっかく作っても使われないのであれば，作る意味はありません．いっとき面倒であっても，コピペで済ませてしまった方がよいことは多々あります．

また，実際に日常の使用に耐えるコマンドを開発するには，引き数の型のチェック，各コマンドのエラー・ハンドリングや，ヘルプ・メッセージの出力などまだまだ考慮すべき点があります．ただ，コマンドを開発することに注意したいことは，UNIXはそれ自体がプログラム環境であり，コマンドを開発する際にもあたかもライブラリを開発しているかのような感覚で開発をした方がよい，ということでしょう．より良いコマンドは多くの便利なコマンドの土台となるので，注意深くコマンドを開発していくことでUNIXはますますパワーアップしていくのです．

なかむら・かずたか

適応処理時代のノイズ・キャンセル実験室

ダウンロード・データあります

新連載 第1回

周波数領域ノイズ除去の基本中の基本…スペクトル・サブトラクション

川村 新

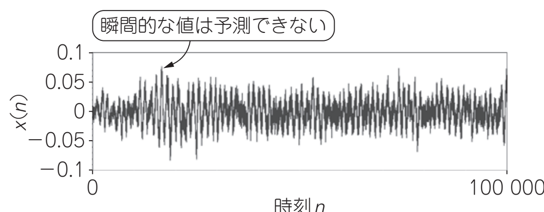


図1　マイクで録音した静かな部屋の音の波形
静かな部屋では主となる音はマイクのノイズになる．各時刻の音を正確に推定することは困難

本連載では音声のノイズ除去に注目し，さまざまな方式や，それらを実現するプログラムを紹介していきます．

今回は，最も基本的な周波数領域のノイズ除去手法であるスペクトル・サブトラクション(Spectral Subtraction)です．

原理

ノイズが含まれる音声をFFT(Fast Fourier Transform；高速フーリエ変換)すると，音声とノイズの和のスペクトルが得られます．そこで，あらかじめ計算しておいたノイズ・スペクトルを減算することで，ノイズを除去します．

● 音声にノイズがのった信号が観測されている

時刻nにおける音声を$s(n)$，ノイズを$d(n)$とし，観測信号$x(n)$を，

$$x(n) = s(n) + d(n)$$

と定義します．ノイズは，環境ノイズに加え，マイクロホン自体の機器ノイズも含まれることとします．

図1は，筆者のノートPCに内蔵されているマイクロホンで，ボリュームを最大にし，環境音を録音した波形です．部屋が静かだったので，主となる音はマイクのノイズです．つまり，

$$x(n) = d(n)$$

です．**図1**から想像できるように，各時刻でノイズの「波形」を正確に推定し，除去することは困難です．

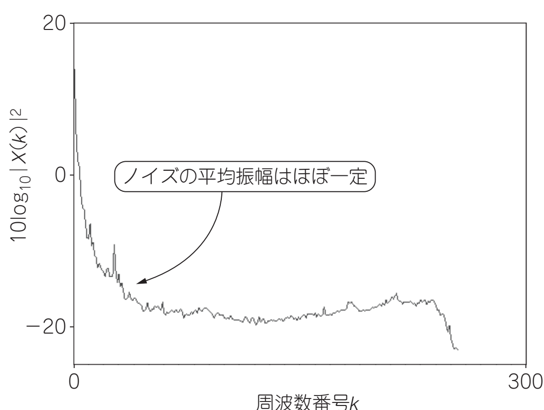


図2　ノイズの平均対数パワー・スペクトル
見やすいように，平均の$|X(k)|$を対数パワー・スペクトルに変換している

● ノイズのスペクトルはほとんど変化しない

観測信号$x(n)$を短い時間間隔(フレームと呼ぶ)でFFTし，スペクトル$X(k)$を調べます．kはスペクトル番号です．$X(k)$は複素数なので，

$$X(k) = |X(k)|\exp(j\angle X(k))$$

と書けます．

ここで，$j=\sqrt{-1}$です．$|\ |$は絶対値で，$|X(k)|$は振幅スペクトルと呼ばれます．$\angle\{\ \}$は偏角で，$\angle X(k)$は位相スペクトルと呼ばれます．

今回は，振幅スペクトル$|X(k)|$のみを調べます．1フレームを32msとして，**図1**の$x(n)$に対して順次FFTを実行し，得られた振幅スペクトル$|X(k)|$を平均した結果を**図2**に示します．

● 観測信号からノイズの推定値を減算する

ノイズだけの観測信号から，あらかじめ平均的な振幅スペクトル$|X(k)|$を取得しておき，これをノイズ推定値$|\hat{D}(k)|$とします．音声とノイズが混在する観測信号から$|\hat{D}(k)|$を減算すれば，ノイズを除去できます．この方法をスペクトル・サブトラクションと呼びます．

k番目のスペクトル$X(k)$に対するスペクトル・サブ

関連特集：本誌2016年6月号「体感! 全集CD付き! 音声信号処理」[1]では，さまざまな音声処理の方式と，それらを実現するプログラムを紹介しています．

トラクションの結果$\hat{S}(k)$は，式(1)のように書けます．

$$\hat{S}(k)=\left(|X(k)|-|\hat{D}(k)|\right)\exp\left(j\angle X(k)\right) \quad \cdots\cdots\cdots\cdots (1)$$

スペクトル・サブトラクションでは，位相スペクトル$\angle X(k)$は処理しません．

● フィルタとして表現する

スペクトル・サブトラクションをフィルタとして表現することもできます．

$$\hat{S}(k)=G(k)X(k)$$

ここで，$G(k)$がフィルタを表しており，スペクトル・ゲインと呼ばれています．式(1)を次のように変形すれば，スペクトル・サブトラクションを実現する$G(k)$が得られます．

$$\hat{S}(k)=\frac{|X(k)|-|\hat{D}(k)|}{|X(k)|}|X(k)|\exp\left(j\angle X(k)\right)$$
$$=G(k)X(k) \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (2)$$

$$G(k)=\frac{|X(k)|-|\hat{D}(k)|}{|X(k)|}=1-\frac{|\hat{D}(k)|}{|X(k)|} \quad \cdots\cdots\cdots\cdots (3)$$

ノイズ除去は，式(3)のスペクトル・ゲインで表現する形式がよく用いられます．

図3に，スペクトル・サブトラクションを実現するブロック図を示します．式(3)の$G(k)$を利用する形で表現しています．ここで，観測信号の最初には音声が存在しないと仮定し，初期数フレームの平均振幅スペクトルを$|\hat{D}(k)|$として利用します．

プログラム

スペクトル・サブトラクションのプログラムの主要部分を**リスト1**に示します．

入力信号はs[t]に格納されます．ここで，tは現在時刻を表します．1フレームのFFT分析に用いるサンプル数FFT_SIZEは512，窓関数w[i]はハニング窓として設定しました．

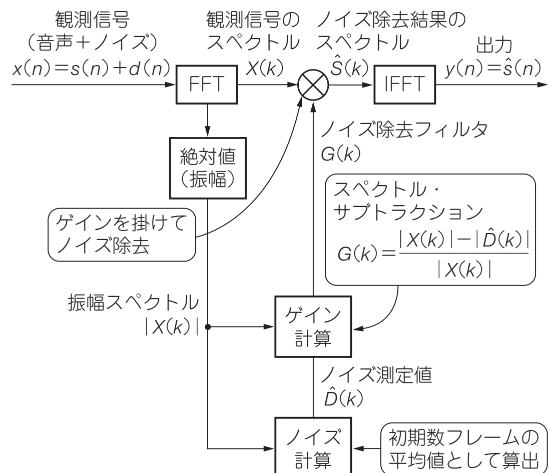


図3　スペクトル・サブトラクションのブロック図

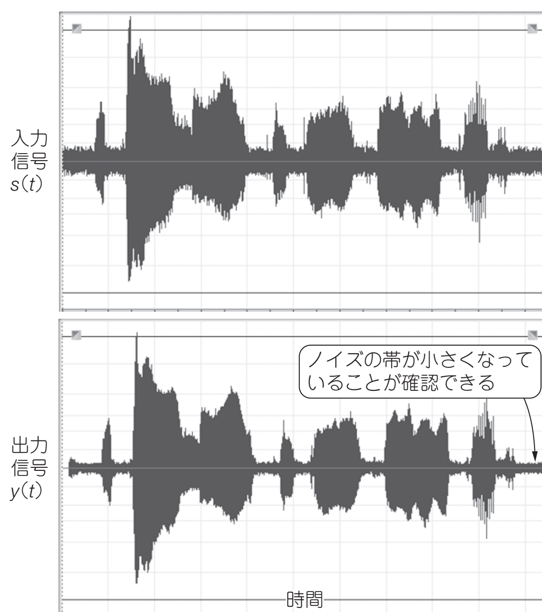
関数fft()を実行すると，窓関数を掛けた入力信号xin[i]に対してFFTを実行し，結果の実部Xr[i]，虚部Xi[i]，振幅スペクトルXamp[i]が計算されます．Xamp[i]を利用してスペクトル・サブトラクションを実行します．

関数ifft()を実行すると，Xr[i]，Xi[i]に対してIFFT(逆フーリエ変換)を実行し，時間領域の信号z[i]を計算します．FFT分析は，FFT_SIZE/2＝256サンプルごとに実行しているため，分析フレームが半分ずつ重複する「ハーフ・オーバラップ」により最終出力y[t]を生成しています．

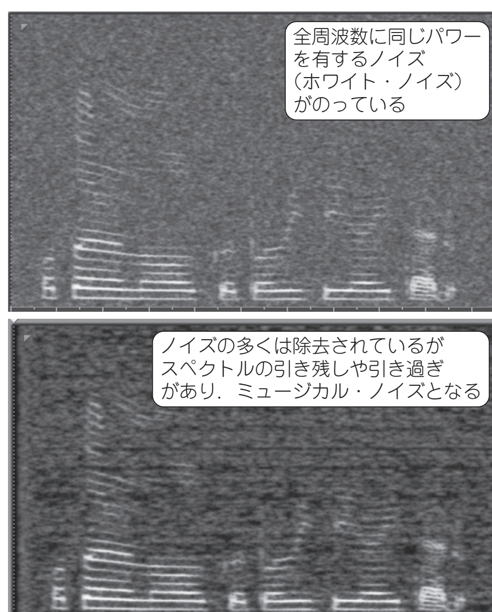
ノイズ推定のためのフレーム数NMを4とし，その平均振幅スペクトルをノイズ推定値lambdaとしました．**図3**における$|\hat{D}(k)|$がlambdaです．さらに，

リスト1　スペクトル・サブトラクションのプログラム(抜粋)

```
while(1){
                                                // メイン・ループ
    if(fread( &input, sizeof(short), 1,f1)<1) break;
                                        // 音声を input に読み込み
    s[t] = input/32768.0;  // 音声の最大値を1とする(正規化)
    x[l] = x1[l];      // FFT_SIZE/2までの入力をx[l]に格納
    x1[l]= x[FFT_SIZE/2+l]=s[t];
                       // FFT_SIZE/2以降の入力をx[l]に格納
    y[t] = (z[l]+z1[l])/FFT_SIZE; // IFFTで得た出力
    z1[l]= z[FFT_SIZE/2+l];
                             // ハーフ・オーバラップ用に出力記録
    if( l>=FFT_SIZE/2){      // 半フレームごとにFFTを実行
        for(i=0;i<FFT_SIZE;i++){
          xin[i]=x[i]*w[i];         // 窓関数を掛ける
        }
        fft();             // FFT
        if(cnt<NM){        // 初期NMフレームの平均がノイズ推定値
        for(i=0;i<FFT_SIZE;i++)
               lambda[i]=lambda[i]+Xamp[i]/(double)NM;
           cnt++;                   // フレーム番号更新
        }
        for(i=0;i<FFT_SIZE;i++){   スペクトル・サブトラクションのゲイン
            G[i]=1.0;
            if(Xamp[i]!=0) G[i]=1.0-lambda[i]/Xamp[i];
            G[i]=( G[i]+fabs(G[i]) )/2.0;
                                       // 負の値をゼロにする
            Xr[i]=G[i]*Xr[i];          // 実部にゲインを乗じる
            Xi[i]=G[i]*Xi[i];          // 虚部にゲインを乗じる
        }
        ifft();                        // IFFT
        l=0;
    }
    l++;                               // FFT用の時刻管理(略)
    output = y[t]*32767;               // 出力を整数化
    fwrite(&output, sizeof(short), 1, f2);
                                       // 結果の書き出し
    t=(t+1)%MEM_SIZE;                  // 時刻 t の更新
}
```



(a) 波形



(b) スペクトログラム

図4　縦軸中心付近に存在する帯状のノイズが小さくなった

スペクトル・ゲインG[i]が負にならないように制限しています．

● 実行結果

スペクトル・サブトラクションの結果を**図4**に示します．入力信号には，ホワイト・ノイズを音声に付加した音声を用いています．

波形から，縦軸中心付近に存在する帯状のノイズが小さくなっていることが確認できます．しかし，試聴してみると，出力にはチャラチャラというような，人工的なノイズが新たに生じていることが分かります．これは，ミュージカル・ノイズ(Musical Noise)と呼ばれるやっかいな音で，スペクトルの引き過ぎや，引き残しによって生じます．ミュージカル・ノイズを完全に除去することは困難で，現在でも研究が続けられています．

● ノイズの推定値を定数倍すると強く除去できる

より強くノイズを除去したいときは，ノイズの推定値を定数倍すると効果的です．これをオーバ・サブトラクションと呼びます．

例として，ノイズ推定値の5倍を減算してみましょう．**リスト1**のプログラム・リスト内の，スペクトル・サブトラクションのゲイン部のG[i]の式を，

```
G[i]=1.0 - 5 * lambda[i]/Xamp[i];
```

に変更してみてください．より強いノイズ除去効果が得られます．ただし，除去性能を強くすると，その代償として音質の劣化が生じやすくなります．

● プログラムはダウンロード提供

本連載で取り上げるプログラムは，すべて本誌ウェブ・サイトからダウンロードできるようにします．また，本誌2016年6月号付録のCD-ROMにも収録されています(スペクトル・サブトラクションのプログラムは17_1_SSフォルダ)．

プログラム名の先頭に「DD」が付いているものは，ドラッグ・アンド・ドロップ・プログラムです．コンパイル後，実行ファイル上に，任意のwavファイルをドラッグ・アンド・ドロップすれば，ノイズ除去後の出力wavファイルが自動生成されます．**図4**の入力信号(noisy_speech.wav)も提供します．

また，先頭に「RT」が付いているものは，リアルタイム・プログラムです．コンパイル後，実行ファイルをダブルクリックすると，マイク入力をノイズ除去処理して，スピーカから出力します．

リアルタイム処理の場合，環境ノイズがなければ効果を確認できないので，今回は，マイク入力に人工的にノイズを付加するプログラムも提供しています．プログラム名の最後に「_noise」が付いているものが該当します．スピーカからは，ノイズ除去後の信号が出力されます．処理，未処理をスペース・キーで切り替えて，ノイズ除去効果を確認してください．Enterキーを押すと，プログラムが終了します．

かわむら・あらた

◆参考文献◆

(1) 川村 新；特集「体感! 全集CD付き! 音声信号処理」，Interface，2016年6月号，CQ出版社．

わりとよく使われるタイプは動かしてガッテン！ ダウンロード・データあります

人工知能アルゴリズム探検隊

第2回 パターン認識でよく使われる「サポート・ベクタ・マシン」

牧野 浩二，渡邉 寛望

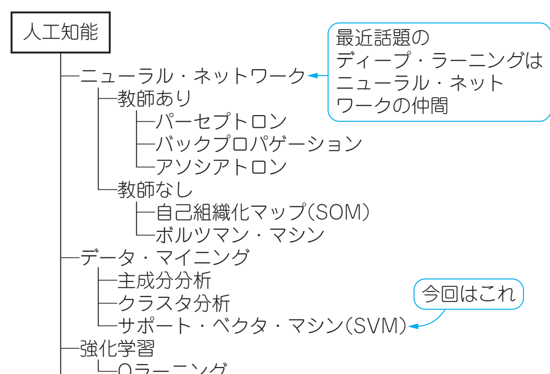


図1　今回の人工知能アルゴリズム…パターン認識でよく使われるサポート・ベクタ・マシン

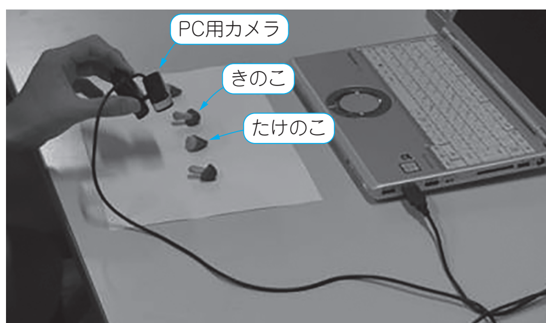


写真1　パターン認識でよく使う人工知能アルゴリズム「サポート・ベクタ・マシン」を用いて「きのこ形のお菓子」と「たけのこ形のお菓子」を分別してみる

● 今回の人工知能アルゴリズム…サポート・ベクタ・マシン

入力したデータがA群に属するのかB群に属するのかをスパッと判断してくれるアルゴリズム「サポート・ベクタ・マシン」を紹介します（**図1**）．今回はこれを利用して，お菓子，「きのこの山」と「たけのこの里」（ともに明治）を分別してみます（**写真1**）．

サポート・ベクタ・マシンは英語で書くとSupport Vector Machineであるため，その頭文字をとって「SVM」と書かれることがよくあります．「マシン」といわれると，車とかロボットとかを想像してしまうかもしれませんが，入力データを二つにうまく分ける（データ・マイニングの）手法の一つです．

サポート・ベクタ・マシンは非常に強力で，パターン認識の分野でよく使われています．応用として，以下があります．

- 指紋認証
- 文字認識
- 人物照合
- ジェスチャ判別
- コンピュータ将棋

また，今後の人間の動作判別や医療データへの応用なども研究されています．

パターン認識向き サポート・ベクタ・マシンの仕組み

● 境界線を機械的に探してくれる

サポート・ベクタ・マシンがやっていることを模式的に表すと**図2**のようになります．ここでは具体的に説明するために，横軸を体重，縦軸を足の長さとし，トラとシマウマのデータを入力します．

サポート・ベクタ・マシンは**図2**に示すように，この2種類のデータをうまく分けるような線を機械的に

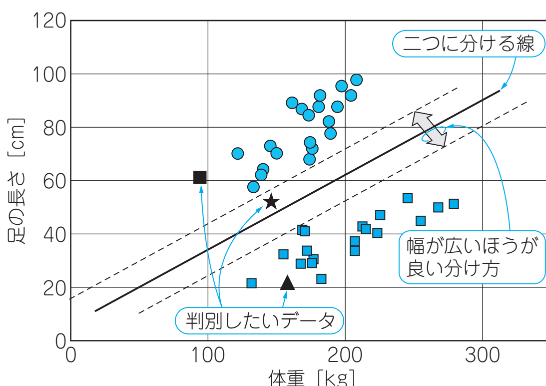


図2　トラ■とシマウマ●のデータを分類

第1回　PICで試せちゃう！なんとなーく思い出せる人工知能記憶装置「アソシアトロン」（2016年9月号）

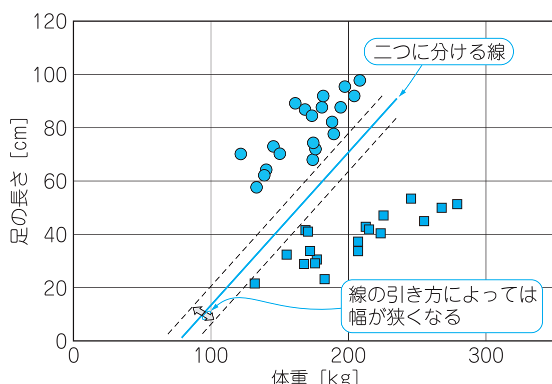


図3　図2はこのように境界線を引くこともできる
悪い境界線の例．トラ：■，シマウマ：●

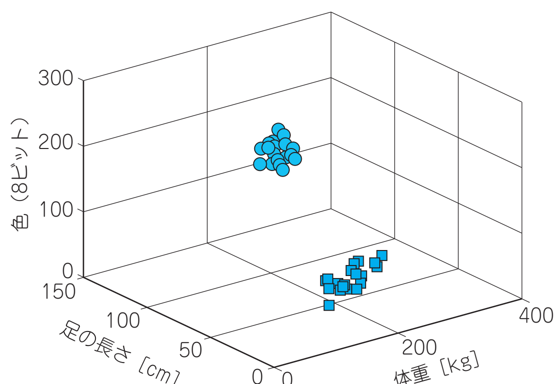


図4　図2に「トラとシマウマの色」を加えれば分類が容易になる
トラ：■，シマウマ：●

探し出すことができます．そして新たなデータを入力したとき，この線より上にあるか下にあるかを判定することで分類します．

例えば，線より上にあればシマウマ，下にあればトラという具合です．線より上か下かで分けるため，サポート・ベクタ・マシンはスパッと二つに分けることができるのです．

● 良い境界線/悪い境界線

ここで**図3**に示すように別の線を引くこともできます．**図2**と**図3**の分け方のどちらがより良い分け方であるかを判定するための基準が必要となります．

サポート・ベクタ・マシンでは，分けるために引いた線から最も近いデータまでの距離が長い方が良い線となります．

図2と**図3**には，分けるための線を平行移動させて最も近いデータまでの距離を点線で表しています．これらの図を比較すると，**図2**の方が良い線となっています．

このように，線の上か下かで判別するため，サポート・ベクタ・マシンはスパっと答えを出してくれます．

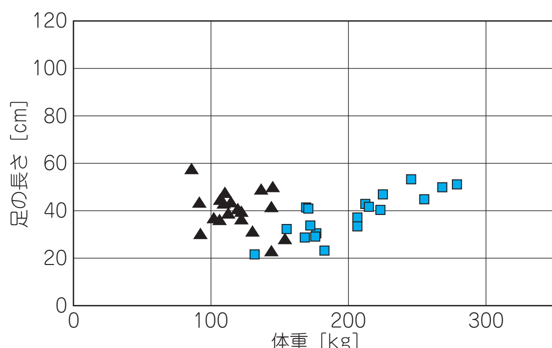


図5　パンダとトラは体重と足の長さだけでは分離しがたい
パンダ：▲，トラ：■

● 境界線を数式で表す

2次元の場合はここまで説明したように1本の線で分けています．どんな線かというと，数学的には次の式で表されています．ただし，この式は本稿だけで詳しく説明することは難しいので割愛します．この境界線は，

$$t_i(\omega\dot{x}_i + \omega_0) \geq 1$$

の制約の下でωのノルムを最小とする解$\min\|\omega\|$として表すことができます．ただし，t_iは1または−1，x_iはi番目のデータ，ωは面の傾きを表す重みベクトル，ω_0は切片に相当するバイアスとします．

上記は2次元に限定する数式ではありません．3次元でも4次元でも何百次元でも成り立ちます．4次元以降は頭の中でイメージするのは難しいのですが，数学上は成り立っています．つまり，特徴となるデータの種類をいくらでも増やすことができます．これが，いろいろな分野で使われる原因となっています．

● 入力するパラメータ（次元）を増やすと分類が容易になる

図2や**図3**では体重と足の長さを入力データとしていました．ここに「トラとシマウマの色」を加えれば，3次元になってしまいますが，**図4**のようにもっと分類が容易になります．

● 分類が難しいときの対処法

しかし，世界中のすべてのデータを二つに分けることはできません．多くの場合複雑に絡み合っています．例えば**図5**のようにトラとパンダを同じ特徴量で分離しようとしても，二つに分けることができない場合があります．

サポート・ベクタ・マシンは，そういう状況でもうまく対処して答えを出せるように，さまざまな手法が考えられています．

その一つにソフトSVMと呼ばれる拡張手法があります．これは，分けられないデータについては，ペナルティをつけて，そのペナルティの合計が小さくなるように線を引く方法です．

その他には，直線でない線で分割するという方法もとられています．

サポート・ベクタ・マシンは日々進化していろいろなデータをうまく分けることができるようになってきています．

実験…きのことたけのこを見分ける!

● 手持ちのPCとフリーのソフトウェアで

それでは実際にサポート・ベクタ・マシンを使ってみましょう．ここでは，**写真1**，**図6**のように，USB接続のPC用カメラで，きのこ形のお菓子とたけのこ形のお菓子を撮影して，スパッと見分けるものを作ります．

そのときのパソコン上の画面は**図7**となります．**図7**(**a**)はカメラの画像，(**b**)は画像処理した後の白黒画像と輪郭，(**c**)がサポート・ベクタ・マシンに加えた入力とその分類結果，(**d**)は見分けた結果となっています．

● 学習と判定の手順

このソフトウェアの使う手順は以下となります．

(**1**) きのことたけのこを一つずつ白い紙の上に置いて，カメラで撮影します．このとき，きのこを撮影するときはキーボードの[K]キーを押し，たけのこを撮影するときは[T]キーを押します．これにより，どちらの画像かをサポート・ベクタ・マシンに教えます．

(**2**) 置き方を変えて(**1**)の撮影を繰り返して教師データを作ります(おおむね30回ずつ)．

(**3**) そのデータを基にして，サポート・ベクタ・マシンで見分けるための直線を求めます．この直線を求めるときには[スペース]キーを押します．

(**4**) その後テストを行うために，(**1**)と同じようにきのこまたはたけのこをカメラで撮影し，[C]キーを押します．そうすると，画面の右下に見分けた結果が表示されます．

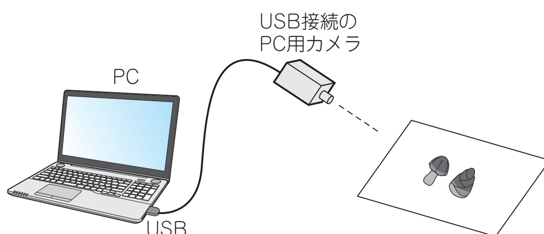


図6　きのことたけのこを撮影してスパッと見分ける装置を作る

● 開発環境

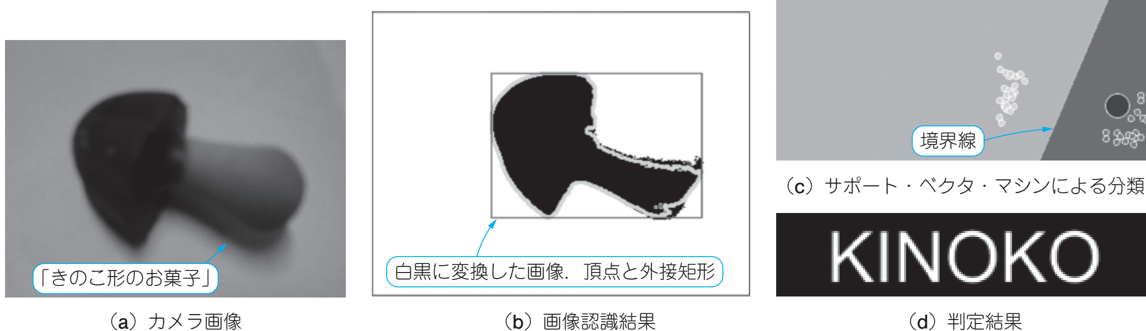
統合開発環境にはProcessingを使います．Processingはプログラムを専門としないデザイナなどが，簡単にパソコン上にきれいな絵を描ける言語として開発されました．その簡単さからさまざまなライブラリが作られて，今回のようなカメラで撮影しながらサポート・ベクタ・マシンを動かすこともできるようになっています．これに以下のライブラリを追加して使います．

• OpenCV

インテルが開発し公開したオープンソースのコンピュータ・ビジョン向けライブラリです．画像処理・画像解析および機械学習などを簡単に使うことができるようになります．

• Blob

画像内のBlob(かたまり)を見つける「ラベリング処理」を行うためのOpenCV用クラスの一つです．かたまりごとに位置や面積，外接矩形などの特徴量を抽出できます．BlobはProcessingに移植されたOpenCVライブラリには実装されていません．そこで別のライブラリとして実装しています．



(**a**) カメラ画像　(**b**) 画像認識結果　(**c**) サポート・ベクタ・マシンによる分類　(**d**) 判定結果

図7　判定した様子

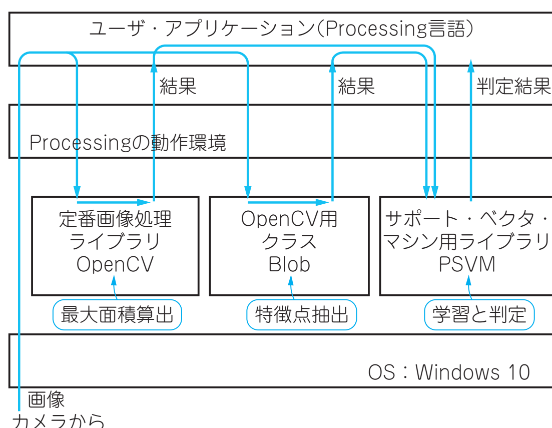


図8　きのこ/たけのこ判定ソフトウェアの構成

- **PSVM**

サポート・ベクタ・マシンのプログラムをProcessingで使うことができるようにしたものです．

ソフトウェアの構成を**図8**に示します．

● 人工知能アルゴリズムへの入力データの準備

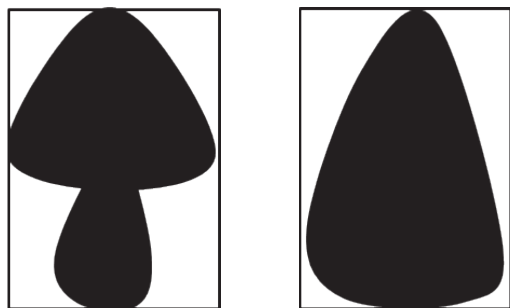
人工知能に関係する手法は入力データをどのように使うのかの前処理が重要となります．今回は画像をそのまま入力として使うのではなく，入力データの以下の特徴量を抽出して，それを入力として使います．

- **頂点の数**

きのこのほうが傘の部分に凸凹が多いため，たけのこに比べて頂点が多くなると予想できます．

- **面積の比**

きのこまたはたけのこを囲む長方形の中にある，「きのこまたはたけのこ」と「背景」との面積比を特徴量として使います．**図9**のようにきのこは柄の部分が細く，たけのこはずんぐりしています．きのこと比べてたけのこは，黒い部分の比率が大きくなると予想できます．



(a) きのこ…白い部分が多い　(b) たけのこ…黒い部分が多い

図9　人工知能アルゴリズムへの入力…面積比

これらを入力として使うのは難しいと感じるかもしれませんが，上記のライブラリを使えば簡単です．

プログラミング

きのこ/たけのこ判定プログラムを**リスト1**(稿末)に示します．ここからは，プログラムの各部の説明を行います．

● 準備…カメラからの画像を取り込む

OpenCVの機能を使ってPC用カメラの画像を取り込んでいます．ProcessingでOpenCVを使うためには次の手順が必要となります．

まず，Processingの「スケッチ」メニューから「ライブラリをインポート」の中の「ライブラリを追加」を選びます．

次にFilterと書いてあるダイアログにOpenCVを入力すると，「OpenCV for Processing」が出てきますので，右下の[Install]ボタンを押してしばらく待ちます．

これでProcessingでOpenCVが使えるようになります．ここでは，OpenCVのサンプル・プログラムのLiveCamTestを改造して使いました．取り込むのは簡単で，**リスト1**の(ア)の`opencv.loadImage(video);`で行うことができ，`image(video, 0, 0);`とすることで画面に表示できます．

● (1) 特徴点を抽出する

Blobを追加することで，簡単にできるようになります．追加の方法はOpenCVと同様で，Blobと入力後に現れる「BlobDetection」をインストールすることで使えるようになります．

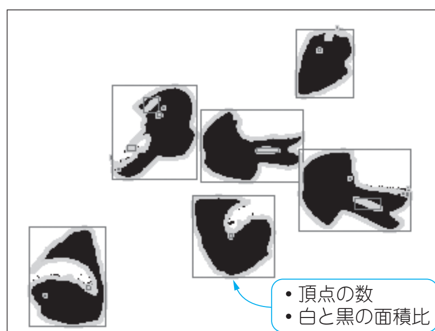
まず，**図7**(b)の頂点ときのこやたけのこを囲む長方形の表示は`drawBlobsAndEdges`関数で行っています．これはBlobのサンプル・プログラムの`bd_image`を改造して使いました．

次にきのこまたはたけのこを囲む長方形の中にある，きのこまたはたけのこと背景の面積比を求めている部分について述べます．これはサポート・ベクタ・マシンへの入力データを作るときに必要となります．

そのため，[K]キーや[T]キー，[C]キーが押されたときに数えるように書かれています[**リスト1**の(イ)]．Blobを使うと自動的に囲む線がいくつか得られます．例えば，きのこやたけのこがたくさんあるような場合には，**図10**のように長方形がたくさん出てきます．

今回は一つだけ映すのですが，一応，得られた長方形の中で一番大きい長方形を対象とするようにしています．そのときの長方形の中にある全てのピクセルに


（a）カメラ画像



（b）判定結果

図10　画像内のかたまりを見つけるためのOpenCV用クラスBlobを用いて面積比を求める

ついて黒かどうかを調べて数えています．これを用いて長方形の面積と黒のピクセルの比を計算しています．

そして，頂点の数を数えている部分について述べます．これもBlobの機能で数えることができ，得られた長方形の中にある頂点を**リスト1**の（ウ）で調べて数えています．

またここで，［K］キーや［T］キーが押されると，**図11**に示すように対応するデータが追加されていきます．このとき，サポート・ベクタ・マシンによる分類が行われる前は，**図7**のように背景が二つに分かれていません．

● (2) サポート・ベクタ・マシンによる分類

▶ライブラリの入手

Processingでサポート・ベクタ・マシンのライブラリを利用するためには，ライブラリをダウンロードして手動で追加する必要があります．

Processing用のサポート・ベクタ・マシンはPSVMというライブラリにまとめられていて，次のサイトからダウンロードできます．

`http://makematics.com/code/psvm/`

ダウンロードしたzipファイルを解凍し，各自のドキュメント・フォルダの中にあるProcessingフォルダの中の`libraries`に移動することで使えるようになります．

▶学習

学習は［スペース］キーが押されたときに行われるため，**リスト1**の（エ）に書かれています．学習データのセットは**リスト1**の（オ）の`problem.setSampleData(labels, trainingPointsNormarize);`で行います．

`labels`と`trainingPointsNormarize`は配列です．`labels`にはそのデータがきのこなのかたけのこなのかを表す1または2という数字が入っていて，`trainingPointsNormarize`には頂点の数と面積比が入っています．ここで注意しなければならないのは入力は0～1までに正規化する必要がある点です．

そして，データをセットしたら**リスト1**の（カ）の`model.train(problem);`によって学習します．学習が終わると**図7**(**c**)のように背景が二つに分かれます．それぞれ30回以上学習させると効果的です．

▶撮影時は背景を白に

このとき，白い紙を敷いて，きのことたけのこだけが映るようにしてください．また，きのことたけのこ



（a）カメラ画像


（b）画像として認識

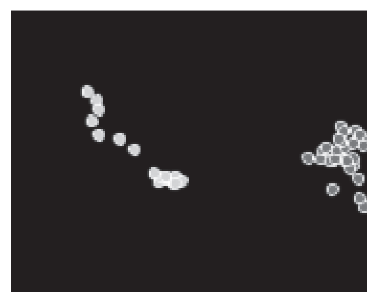

（c）頂点の数を求めた

図11　Blobを用いて頂点数を求めているようす

リスト1　きのこ/たけのこ判定プログラム

```
import gab.opencv.*;
import processing.video.*;
import java.awt.*;
import psvm.*;
import blobDetection.*;

SVM model;          //サポート・ベクタ・マシンを使うための宣言
BlobDetection theBlobDetection;
//外接矩形などを使うための宣言
SVMProblem problem;
//サポート・ベクタ・マシンの学習のための宣言
OpenCV opencv;      //OpenCVを使うための宣言

Capture video;      //カメラ画像をキャプチャするための宣言
PImage src, dst;    //カメラ画像と白黒変換画像の保存
PImage img;         //Blob画像の処理用

float[][] trainingPoints;
//学習データ：2次元（エッジの数，白黒の割合）
int[] labels;               //ラベル（1：きのこ，2：たけのこ）
int num;                    //学習データの数
float[] checkp;             //テスト・データ：2次元
int checkn;                 //テスト・データのラベル
float maxcontours = 0;      //エッジの数の最大値

PGraphics modelDisplay;

void setup() {
  size(640, 480);        //ウインドウのサイズ
  video = new Capture(this, 640/2, 480/2);
  //カメラ・キャプチャの画像用
  opencv = new OpenCV(this, 640/2, 480/2);
  //OpenCVで扱う画像用
  modelDisplay = createGraphics(640/2, 480/2);
  //サポート・ベクタ・マシンの結果表示部の画像用
  img = new PImage(640/2, 480/2);   //Blob処理画像用
  theBlobDetection = new BlobDetection(img.width,
                       img.height);       //Blobの設定
  theBlobDetection.setPosDiscrimination(true);
  theBlobDetection.setThreshold(0.2f);   //閾値

  num = 0;          //学習データ数の初期化
  trainingPoints = new float[200][2];
  //学習データ保存用（最大200個まで）
  labels = new int[200];         //ラベル保存用（最大200個）
  model = new SVM(this);         //SVM処理用
  problem = new SVMProblem();    //SVM処理用
  problem.setNumFeatures(2);     //SVM処理用
  checkp = new float[2];         //テスト・データの初期化
  checkp[0] = -10;
  checkp[1] = -10;
  checkn = 3;
  video.start();    //ビデオ・キャプチャ・スタート
}

//Blobによるエッジと外接矩形の抽出用関数
void drawBlobsAndEdges(boolean drawBlobs, boolean
                                          drawEdges)
{
  noFill();
  Blob b;
  EdgeVertex eA, eB;
  for (int n=0; n<theBlobDetection.getBlobNb(); n++)
  {
    b=theBlobDetection.getBlob(n);
    if (b!=null)
    {
// エッジの抽出
      if (drawEdges)
      {
        strokeWeight(3);
        stroke(0, 255, 0);
        for (int m=0; m<b.getEdgeNb(); m++)
        {
          eA = b.getEdgeVertexA(m);
          eB = b.getEdgeVertexB(m);
          if (eA !=null && eB !=null)
            line(
              eA.x*width/2, eA.y*height/2+height/2,
              eB.x*width/2, eB.y*height/2+height/2
              );
        }
      }
//外接矩形の抽出
      if (drawBlobs)
      {
        strokeWeight(1);
        stroke(255, 0, 0);
        rect(
          b.xMin*width/2, b.yMin*height/2+height/2,
          b.w*width/2, b.h*height/2
          );
      }
    }
  }
}

//サポート・ベクタ・マシン表示用関数
void drawModel() {
//サポート・ベクタ・マシンの結果を表示
  modelDisplay.beginDraw();
  modelDisplay.background(0);
  if (num>0) {
//右上のすべてのピクセルについてサポート・ベクタで分類し色分け
    for (int x = 0; x < width/2; x++) {
      for (int y = 0; y < height/2; y++) {

        //画像の位置からテスト・データを作成
        double[] testPoint = new double[2];
        testPoint[0] = (double)x/(width/2);
        testPoint[1] = (double)y/(height/2);

        //テスト・データを分類したラベルを変数dに
        double d = model.test(testPoint);

        //ラベルによって色分け
        if ((int)d == 1) {
          modelDisplay.stroke(255, 0, 0);
        } else if ((int)d == 2) {
          modelDisplay.stroke(0, 255, 0);
        } else if ((int)d == 3) {
          modelDisplay.stroke(0, 0, 255);
        } else {
          modelDisplay.stroke(255, 255, 255);
        }

        //設定した色で点を打つ
        modelDisplay.point(x, y);
      }
    }
  }
//結果の書き込みの修了
  modelDisplay.endDraw();
}

void draw() {
  scale(1);
  background(0);
```

の白黒画像があまりきれいに出ない場合は**リスト1**の（キ）のopencv.threshold(70);の引き数を0～255までの範囲で変えてください．

▶分類結果の出力

そして最後に，テスト・データを入力して分類結果を出力します．これは[C]キーが押されたときに行われるため**リスト1**の（ク）に書かれています．

データの取得方法は同じなので，大部分は[K]キーや[T]キーが押されたときと同じです．テスト・データの分類は**リスト1**の（ケ）のcheckn = (int)model.test(p);によって行われます

この戻り値はデータ・ラベルとなり，きのこと分類されれば1が，たけのこであれば2となります．そして，この値によって**図7**(d)のように表示されます．

```
  opencv.loadImage(video); //カメラ画像の取得          (ア)
  image(video, 0, 0 );     //カメラ画像を左上に表示
  opencv.gray();           //二値化
  opencv.threshold(70); ←(キ)
  dst = opencv.getOutput();

  image(modelDisplay, 640/2, 0);
  //右上にサポート・ベクタ・マシンの分類結果を表示
  strokeWeight(1);

//サポート・ベクタ・マシンの学習データの表示
//小さな丸印で表示
  stroke(255);
  for (int i = 0; i < num; i++) {
    if (labels[i] == 1) {
      fill(255, 0, 0);
    } else if (labels[i] == 2) {
      fill(0, 255, 0);
    } else if (labels[i] == 3) {
      fill(0, 0, 255);
    }
    ellipse(trainingPoints[i][0]/maxcontours*width/
      2+width/2, trainingPoints[i][1]*height/2, 5, 5);
  }

//サポート・ベクタ・マシンのテスト・データの表示
//大きな丸印で表示
  if (checkn == 1) {
    fill(127, 0, 0);
  } else if (checkn == 2) {
    fill(0, 127, 0);
  } else {
    fill(0, 0, 127);
  }
  ellipse(checkp[0]*width/2+width/2, checkp[1]*height/
                                          2, 20, 20);

  image(dst, 0, 480/2);  //右上にSVMの表示

//外接矩形とエッジを見つけるためのBlob処理
  img.copy(video, 0, 0, video.width, video.height,
    0, 0, img.width, img.height);
  theBlobDetection.computeBlobs(img.pixels);
  drawBlobsAndEdges(true, true);

//右下に判定結果を表示
  PFont font;
  font = createFont("Arial", 36);
  textFont(font);
  fill(255);
  if (checkn==1) {  //きのこの山ならば
    text("KINOKO", width*3/5, height*3/4);
  } else if (checkn==2) {  //たけのこの里ならば
    text("TAKENOKO", width*3/5, height*3/4);
  }
}

void captureEvent(Capture c) {
  c.read();
}
                                                  (エ)
void keyPressed() {
  if (key == ' ') {  //スペース・キーが押されたとき
//保存した学習データからSVM用のデータを作る
    float[][] trainingPointsNormarize = new float
                                        [num][2];
    for (int i = 0; i < num; i++) {
      trainingPointsNormarize[i][0] = trainingPoints
                              [i][0]/maxcontours;
      trainingPointsNormarize[i][1] = trainingPoints
                                          [i][1];
    }
    problem.setSampleData(labels,
trainingPointsNormarize);  //データを設定する ←(オ)
    model.train(problem);  //SVMの学習を実行 ←(カ)
    drawModel();
  } else if (key == 'k' || key =='t' || key =='c') {
                         //k、t、cキーが押されたとき
    float[] p = new float[2];                    (イ)
    Blob b;
    EdgeVertex eA, eB;
    float s = 0;
    int e = 0;
    int k = 0;
//外接矩形の中で最大の面積となるものを探す
//blobは複数の外接矩形を一度に探すことができる
//複数あった場合は一番大きいものを対象とする
    for (int n=0; n<theBlobDetection.getBlobNb(); n++)
    {
      b=theBlobDetection.getBlob(n);
      if (b!=null)
      {
        if (s<b.w+b.h) {
          s=b.w*width*b.h*height;
          //最大となる外接矩形の大きさを計算する
          e=b.getEdgeNb();
          //その中のエッジの数を調べると ←(ウ)
//その中の白と黒の割合の計算
          k = 0;
          for (int i = 0; i < (int)(b.w*width); i++) {
                                   //全ピクセルを探索
            for (int j = 0; j < (int)(b.h*height);
                                             j++) {
//黒ならば(二値化画像なので、赤要素が0であることを調べればよい)
              if (red(dst.get(i+(int)(b.xMin*width),
                    j+(int)(b.yMin*height)))==0)
                k++;
            }
          }
        }
      }
    }
    p[0] = e;          //エッジの数
    p[1] = k/s;        //外接矩形の面積に占める黒の割合
    if (key=='c') {    //cキーが押されたとき
      checkp = p;      //テスト・データを作成
      p[0]/=maxcontours;                          (ク)
      checkn = (int)model.test(p);(ケ)
      println("check: " + checkn);
    } else {
      if (maxcontours<e) {
        maxcontours = e;
      }
      trainingPoints[num] = p;
      if ( key == 'k' ) {  //kキーが押されていたら
        labels[num] = 1;
        //ラベルを1にする(きのこであることを示す)
      } else if ( key == 't' ) {  //tキーが押されていたら
        labels[num] = 2;
        //ラベルを2にする(たけのこを示す)          (イ)
      }                                          ここまで
      num++;  //学習したデータの数
    }
    println("area k: " + k + " s:" + s + " k/s:" + k/
          s + " e:" + e/maxcontours + " contours");
    //デバッグ用にシリアル・モニタに表示
  }
}
```

たけのこの場合は「TAKENOKO」と表示されます．

＊　＊　＊

今回はサポート・ベクタ・マシンの簡単な原理と使い方を紹介しました．サポート・ベクタ・マシンは強力なツールですが，その前処理の重要性も紹介できたのではないでしょうか．

まきの・こうじ，わたなべ・ひろみ

スポーツ好き向けオープンソース！
動作解析から撮影のコツまで

運動映像解析ソフトKinovea



清水 潤

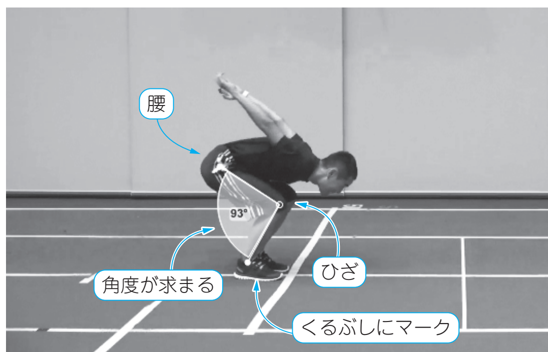


図1 オープンソース・ソフトウェアKinoveaは運動映像から動作を解析できる
関節の中心にツールの点を配置する

オープンソース運動映像解析ソフトKinovea

計測と聞けばセンサを思い浮かべる方が多いと思いますが，映像を利用してさまざまな計測を行うことができます（**図1**）．映像を利用した計測は，大きく分けると2次元的な計測と3次元的な計測に分けられます．2次元的な方法では，運動がある平面で行われていると仮定して，1台のカメラで撮影し分析を行います．一方，3次元的な方法では，複数台のカメラで撮影を行い，身体部位や用具の3次元座標を算出して動作の分析を行います．

表1 Kinoveaの動作環境

項目	内容
OS	Windows（XP, Vista, 7, 8, 8.1）
実行環境	.NET Framework 2.0以降
CPU	1GHz
メモリ	256Mバイト
スクリーン・サイズ	1024×600ピクセル
入力ビデオ・フォーマット	AVI, MPG, MOV, WMV, MP4, FLV, 3GP, MKV, VOB, MOD, TOD
入力ビデオ・コーデック	DV, DivX, Xvid, H.264, MJPEG, Theoraなど

3次元的な方法は測定がより正確にできるメリットがありますが，作業量が格段に増えるというデメリットがあります．2次元的な分析では，ある平面上での運動と仮定するため制限がありますが，簡単に行うことができるというメリットがあります．

今回はKinoveaという無料の映像分析ソフトウェアを例に映像を利用したスポーツ計測における撮影のコツを紹介していきます．

● 基本機能

Kinoveaは映像から2次元動作分析を行うアプリケーションで，距離，角度，速度の計測ができます．また，映像中にコメントや軌跡の描画，二つの映像を同時に再生して比較，これらの映像の書き出しもできます．

● 動作環境

KinoveaはWindows（XP, Vista, 7, 8, 8.1）上で動作するアプリケーションで，.NET Framework 2.0以降がインストールされている必要があります．詳しくは**表1**を参照してください．安定版はver. 0.8.15ですが，日本語メニュー表示をしたい場合はver. 0.8.24をインストールする必要があります（`http://www.kinovea.org/en/downloads/`）．

● 基本操作

映像を選択後，映像下に表示されるツール群を利用して各種作業を行います（**図2**）．測定したいシーンを再生またはコマ送りで探します．測定したいツールを選択（クリック）して，映像上に各種ツールを描画してみましょう．

▶距離の計測

ライン・ツールをクリックすることで，ライン・ツール・モードになります（**図3**）．映像上の線を引きたい部分の始点でマウスダウンし，そのままドラッグして終点でマウスアップします．なお，ライン・ツールのアイコンを左クリック長押しすると他のツールも表示されます．

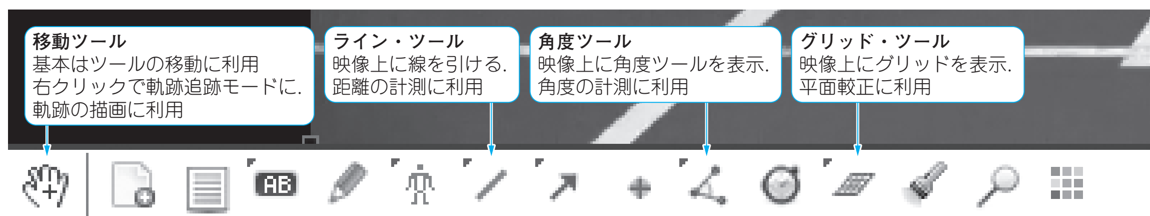


図2　オープンソース運動映像解析ソフトKinoveaはマウス操作で距離や角度を測れる

▶**角度の計測**

角度ツールを選択後，映像上の任意の位置でクリックします．角度ツールが表示されます(**図4**)．角度を測定したい部位にツールの3点を配置します．なお角度ツール・アイコンを左クリック＆長押しで他の角度に関するツールを選択できます．

▶**軌跡の描画**

移動ツールがアクティブになった状態で，映像上の任意の場所で右クリックします．メニューが表示されるので，「軌道追跡」を選択します．軌道を追跡したい部分に十字を合わせます．後はコマ送りしていけばKinoveaが自動追尾を始めます．ずれた場合は手動で調整します．

▶**速度の計測**

距離のキャリブレーション，軌跡の描画が完了後，軌跡上で右クリックするとメニューが表示されるので「Data analysis」を選択します．ポップアップが表示されます．「Data source」から速度や加速度を選択できます．グラフやデータだけを出力することも可能です．

運動映像解析のためのカメラ設定のコツ

ビデオ・カメラではさまざまなパラメータを設定できます．これらを上手に設定して計測しやすい映像を撮影します．

● フレーム・レート…速い動作はなるべく高めに

一昔前のビデオ・カメラでは，フレーム・レートを選択することができませんでしたが，最近のビデオ・カメラでは，60fpsなどと設定できます．iPhone6以降では240fps，Exilim(カシオ)では最高1000fpsでの撮影も可能となっています．

ラケットやボールなど高速で移動するものを測定対象としたい場合，フレーム・レートが遅すぎると1コマ間での移動距離が大きくなってしまい，測定精度が落ちてしまいます．速い動作を対象としたい場合は速いフレーム・レートで撮影するとよいでしょう．

Kinoveaではハイスピードで撮影した映像にも対応しています．当然のことですが，フレーム・レートを速くすると1秒間当たりのコマ数は増えていくので，計測作業量が増えていきます．動作の速さに合わせて最適なフレーム・レートを模索してみてください．

● シャッタ・スピード…速い方がぶれが少ない

映像からさまざまな測定を行う場合，コマ送りをして画面上の部位の位置特定をコマごとに行います．このとき，シャッタ・スピードが十分速くないと動作を止めることができず，映像を止めた静止画においてぶれたものになってしまいます．**写真1**(a)は1/60秒，**写真1**(b)は1/1000秒のシャッタ・スピードで撮影したものです．ボールのパスにおいては1/60のシャッ

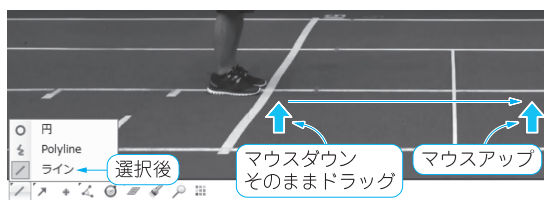
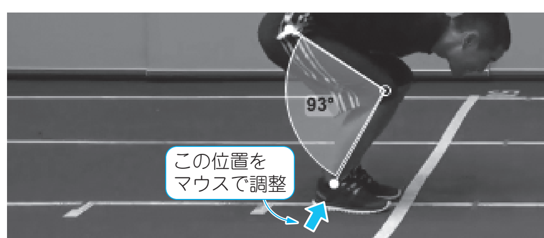


図3　距離の計測はマウスのダウンとアップで



(a) 角度ツールを選択



(b) 測定したい部位にツールを配置

図4　角度計測ツールの利用で膝の曲げ具合などが求まる



（a）シャッタ・スピード1/60秒



（b）シャッタ・スピード1/1000秒

写真1　シャッタ・スピードの違いによる動作の静止具合

タ・スピードではボールがぶれてしまい正確な位置特定ができません．より正確な位置特定を行えるよう，動作を十分止めることができる速さのシャッタ・スピードで撮影を行いましょう．なお，シャッタ・スピードを速くすればするほど映像は暗くなっていくので，特に光量が少なくなる屋内ではISO感度や絞りを調節して明るさを確保してください．

● 走査…インターレースよりプログレッシブ

撮影した映像をコマ送りしたらギザギザした静止画となってしまい，見づらかった経験はないでしょうか．テレビでのなめらかな再生を目的としてインターレース技術が発展してきましたが，パソコンなどのモニタはプログレッシブ走査のため，インターレースで撮影された映像はギザギザした映像となってしまいます．60iなどインターレースで撮影された映像では1/120秒ずれた奇数フィールドと偶数フィールドが表示されるため，結果としてギザギザとした静止画になってしまいます(**写真2**)．映像から計測を行うときにはこのようなギザギザな映像では位置特定の妨げとなります．このようなことが起きないよう60pなど，プログレッシブ・モードで撮影を行いましょう．

運動映像解析のための撮影のコツ

何を計測したいか，どんな平面の運動を捕らえるかをしっかりと考えることが後の計測精度の向上につながります．Kinoveaでの2次元計測では運動面が垂直平面か水平平面のどちらかになります．

● 位置や倍率を固定する

パン，チルトなどカメラを動かすと運動の行われている平面が映像中で移動することになり，コマごとにキャリブレーションが必要となります．2次元的な分



（a）インターレース



（b）プログレッシブ

写真2　走査方式の違い

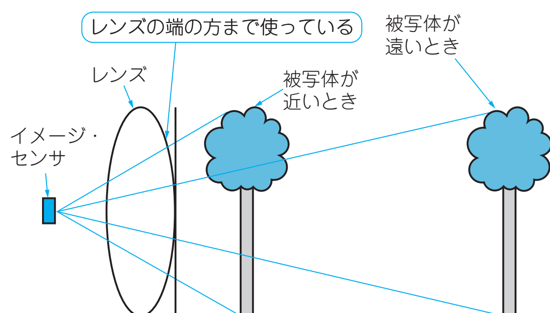


図5　ワイド寄りだとより直角の広いレンズを使用するため周囲がゆがむ

析を行う場合は，カメラを固定して撮影することをお勧めします．もちろんズームも固定します．

● 目的の動作がすべて収まるように撮影する

計測したい部位，道具がきちんと映像に収まるように撮影します．テレビ放送などでは演出もあり身体の一部だけがアップで撮影されることが多くありますが，計測を目的とした場合は計測したい部分が全て撮影されるように注意します．例えばゴルフのスイングを計測したい場合は，試し撮りを行いながら，全てのスイング動作とゴルフ・クラブが見切れることのないようなサイズで撮影します．

● できるだけ遠くからズームして撮影する

カメラの光学ズームを利用できる場合は，可能な限り被写体から離れた位置にカメラを設置してズームを利用して撮影します．同じものを画面上で同じサイズになるようにワイド側とテレ側で撮影した場合，ワイド側の撮影ではレンズの中央より離れた部分を利用することとなり，結果として周辺部分がゆがむことになります．ゆがみが生じてしまうと，長さの換算が正しくできません．カメラの位置を被写体から離してズームを使うことで，よりレンズの中央を使うことができ，結果としてゆがみを減らすことができます(**図5**)．

また，同じ理由により，被写体をできるだけ画面の中央に収まるように撮影することが望ましいと言えます．

● カメラの位置を考える

垂直平面を切り取る場合，作業を減らすことや計測精度を上げるために運動面とカメラの光軸とが垂直になるようカメラ位置を決定します(**図6**)．ここでの運動面とは，走っている運動方向の直線とその鉛直方向を含む面のことです．この面にカメラの光軸を直交させることで，この平面上のどの位置でも距離を同じく扱うことができるようになります．もしこの面とカメラの光軸が直交していなかった場合，この平面上の同じ長さでもカメラから遠い部分は映像上で短く，近い部分は長く表現されてしまいます(**図7**)．これでは運動面上の計測を同じ条件で行うことができません．運動面とカメラの光軸が直交していなくても距離の補正を行える2次元DLT法という手法もありますが，計算が複雑になり手間が増えます．簡易的にすぐ結果を出せるように運動面とカメラの光軸は直交させるとよいです．なお，Kinoveaでは水平平面に対してカメラの光軸が直交していなくても補正する機能があります．

水平平面上での運動をとらえたい場合は，競技場などではスタンドの最上部からなど，できるだけ高い位置から撮影します．水平面をカメラで捕らえるとき，撮影位置が低いと奥行きを画面上で広く表現することができなくなってしまい，結果として奥行き方向の計測精度が下がってしまいます．

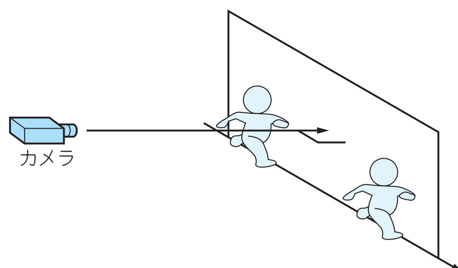


図6　撮影時はカメラの光軸と運動面(垂直)を直交させる

● マーキングをする

ビデオを使った計測では，計測したい身体部位や道具の位置をデジタイズする作業が伴いますが，このデジタイズ精度が低いと，計測結果も精度が低くなってしまいます．デジタイズ精度を高くし，またデジタイズ作業を楽にするために，計測したい身体部位や道具にシールやテープなどでマーキングした状態で撮影します．このとき，身に付けている衣類や背景を考慮して，目立つ色を利用するとよいでしょう．

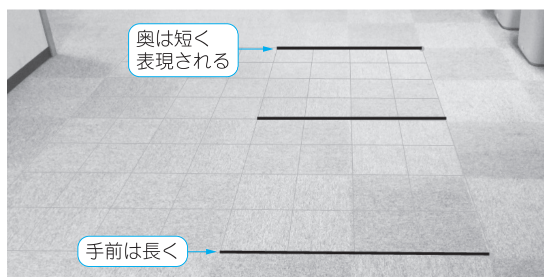


図7　奥行きと画面上の長さの関係

● スマートフォンやタブレットで撮影するときはデバイスの向きに注意する

最近のスマートフォンやタブレットのカメラ性能の向上は著しく，ビデオ・カメラを利用しなくても事足りるだけの性能を持ち合わせてきました．スマートフォンなどでの撮影は手軽で便利ですが，撮影時のデバイスの向きが重要になってきます．例えばiPhoneの場合，iPhoneで撮影された映像はQuickTime形式で撮影され，QuickTime Playerを利用した再生では撮影時のデバイスがどんな向きでも再生時に自動で画面を回転させ正しく再生してくれます．しかし，他のアプリケーションで再生を行うと映像の向き情報が正しく解釈されないため，結果的に逆さまの映像や90°傾いた映像となってしまいます．このようなことを避けるため，iPhoneの場合ではデバイスを横方向にし，ホーム・ボタンが右にくるようにして撮影すると，正しい向きの映像を撮影できます．他のデバイスを利用するときはデバイスをどの向きにして撮影すべきかテスト撮影をしましょう．

機能1…距離の計測

パフォーマンスの最終結果として距離が大切となる動作はたくさんあります．その距離の計測をビデオから行うことができます．長さの基準となるものを一緒に撮影することでKinoveaを利用できます．



図8　距離計測用のキャリブレーション

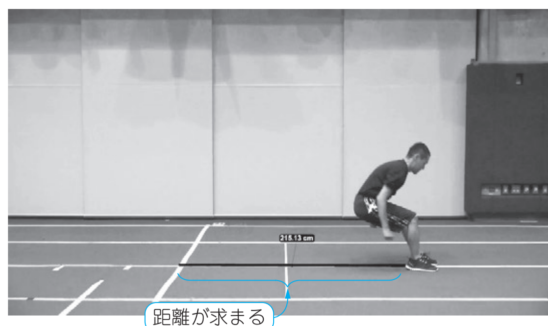


図9　踏切位置から着地位置まで線を引くだけで跳躍距離を算出

● 垂直平面の運動の距離を計測する場合

▶物差しを一緒に映す

映像を使って距離を計測するためには，映像中の長さを実世界の長さに変換する必要があります．具体的にはあらかじめ長さが分かっているもの，または長さが分かっている位置にマーキングを行い，これを一緒に撮影します．そうすると，あらかじめ長さが分かっているものが映像上では何ピクセルあるかを計算することにより，画面上の長さを実際の長さに変換できます．

長さの基準を設定する際に注意すべきことがあります．この長さの基準は運動面上に配置します．映像は奥行きを上手に表現できないため，同じ長さのものが奥に行けば短く，手前に来れば長く表現されます（**図7**）．そのため，物差しとなる長さの基準が運動面上にないと物差しが狂ってしまい，測定される値が大きくずれる原因となります．今回の幅跳びの例では，踏切と着地を結ぶ直線上に長さの基準を配置することになります．

▶キャリブレーション

まず実世界の長さと映像中の長さを変換するための基準を作ります．具体的にはKinoveaで撮影された物差し部分に線を引き，その線の実世界での長さを設定します（**図8**）．ここでは長さの単位も設定可能です．

▶いざ計測

いよいよ距離の計測を行います．Kinoveaには先ほどのキャリブレーションで画面上の何ピクセルが何センチであるか設定されたため，画面上にライン・ツールを利用して線を引くだけで距離の測定ができます．立ち幅跳びの場合は，踏切位置，踏切時のつま先位置から着地位置，着地したかかと位置までライン・ツールで線を引きます（**図9**）．この作業を行うだけで，跳躍距離を計測できます．なお，撮影時に踏切位置をマークしておくとこの線を引きやすくります．撮影時に踏切位置をマークできなかった場合は，映像中の踏切位置にマークをつけておくと作業が楽になります．

● 水平平面の運動の距離を計測する場合

▶できるだけ高い位置から　基準となるものが最低でも3点入るように撮影する

前述しましたが，できるだけ高い位置から撮影を行いましょう．その際に，サッカーやバスケットボールなどでは，サイズが決まっているコートを，上手に映像に収めます．コート上の角やラインなど最低でも3点が映像に収まるようにしてください．

グリッド・ツールのラインとコートのラインを合わせることで4点目を推定できます．コートのように距

離が分かるものがない場合は，運動を行うフィールド上に長方形を定義し，その角にマーキングを行います．この場合は4点をしっかりと映像に収めます．

▶キャリブレーション

Kinoveaのグリッド・ツールを利用して平面較正を行います．グリッドの角をコートの角やマークの角に合わせます．その後，その長方形の短辺，長辺の長さを設定画面で入力すれば較正設定は完了です．例えばフットサル・コートの場合はグリッドをコートに合わせ，平面較正で縦横のコート・サイズを設定します．

▶計測

キャリブレーション設定が完了後，ライン・ツールで線を引くだけでグリッド内の距離，例えば選手間の距離を測ることができます．ここでの注意点は，この計測において高さのあるものを基準にすることができないことです．例えば選手の頭を基準にした距離の測定は，大きな測定誤差を産む原因となります．選手の両足の真ん中，体の重心から真下に下ろした線とコートである平面がぶつかる点をイメージし，その点同士を線で結びます(**図10**)．

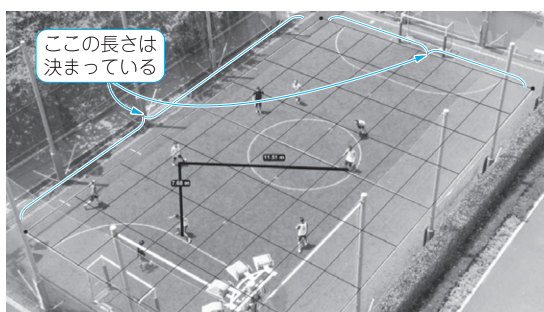


図10　グリッドで平面較正すると選手間の距離が計測できる

機能2…角度の計測

関節の角度やボールの飛んでいく角度など，パフォーマンスを評価する際に角度の把握が必要となることがあります．そんな角度を映像から計測していきます．

● 垂直平面の運動の角度を計測する場合

▶抽出したい動きを精査し撮影方向を決める

評価したい動きはどの方向から撮影すべきか考える必要があります．身体運動において1平面内で完結できるような単純な動作は少なく，3次元的に動作するものが大多数と言えます．例えば肩はかなり複雑な動きをしており，2次元でとらえられる事象は，肩の動きのほんの一部に過ぎません．Kinoveaでの角度の計測では2次元の1平面を切り取ってその方向から見える角度だけを計測することになるので，どの方向から撮影するかが重要になります．

角度の計測では，3点を結ぶ直線が挟む角度を測ることになりますが，この3点の位置をしっかり決めないと，計測のたびに値が変わってしまいます．こうしたことが起きにくいように，計測したい部位，関節の中心にシールなどでマーキングをしておくと，後の作業で位置特定が楽になります．

▶キャリブレーション

キャリブレーションのための特別な作業はありません．しっかりと水平が取れた状態で撮影します．

▶計測

Kinoveaの角度ツールを選択し，目的の部位上でクリックします．分度器のようなツールが表示されるので，計測したい関節など，**図1** (p.114) の場合は腰，ひざ，くるぶしの位置にツールを配置します．マーキングをしている場合はマークを利用して，マーキングをしていない場合は骨格がどのようになっているかをイメージしながら，体勢によって関節の中心がどこにあるかを判断します．3点の位置が姿勢によって，またはコマによって異なってしまうと関節角度を正しく計測できなくなってしまいます．

● 水平平面の運動の角度を計測する場合

▶位置基準点を3点以上撮影する

水平平面の距離を計測する場合と同じで，できるだけ高い位置から撮影し，コートなどキャリブレーションに利用できるマークがしっかりと入るようにします．

▶キャリブレーション

前項の「距離測定→水平平面」と同じく，グリッドをコートやマークに合わせて配置し，距離を設定します．菱(ひし)型となったグリッドを設定した長さの長方形として補正し，角度を計算してくれます．

▶計測

キャリブレーションで配置したグリッド上で角度ツールを利用して角度の計測を行います．補正された値が表示されます(**図11**)．

機能3…軌跡の描画

計測とは少し異なりますが，映像上に軌跡を描画する方法を紹介します．関節などの部位や，ラケットやボールなどの軌跡を画面上に表示できると，部位や道具の動きを全体的に把握できます．パフォーマンスを評価したい部位や道具の軌跡を描画してみます．

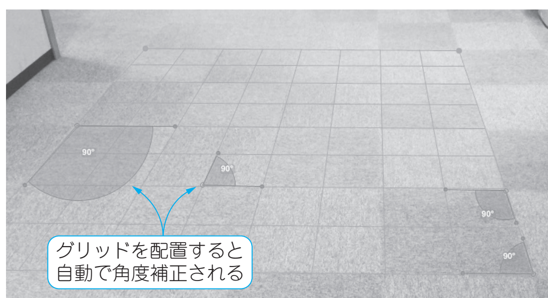


図11　水平平面上にグリッドを配置することで角度のキャリブレーションができる

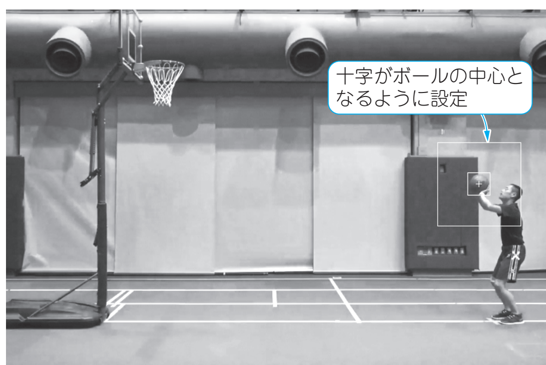


図12　ボールの中心に十字を合わせる

● 軌跡の描画を目的とした撮影の注意点

▶背景を考える

関節部位や道具の軌跡描画を目的とした撮影を行う場合，被写体の背景を考慮すると後の作業が楽になります．被写体と背景が同系色であったり，背景が複雑なものであったりすると，軌跡を引きたい部位の特定が難しくなります．Kinoveaでは設定した対象物を自動で追従する機能がありますが，対象物と背景とのコントラストがはっきりしていないと正確に自動追従してくれません．他のソフトウェアなどで同じことをする場合でも，背景に注意を払うと後の作業が楽になります．

▶どのような方向からの撮影がよいか考える

同じ動作でも撮影する方向によって見える動きは異なってきます．軌跡を描くことで何を見たいのかをイメージし，撮影方向を決めます．

▶マーキングする

可能であれば軌跡を描きたい部位にシールなどでマーキングをしておくと，画面上での部位の位置特定をしやすくなります．その際，部位の中心，道具の中心にマーキングするとよいでしょう．

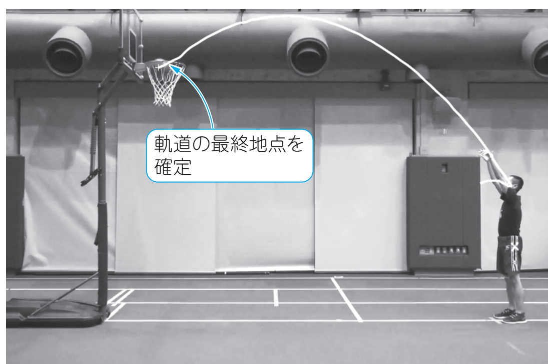


図13　フリースローでのボールの軌跡

● いざ軌跡を描く

軌跡を描きたい部位，道具を画面上で設定します．今回はバスケットボールのフリースロー時のボールの軌跡を描いてみます．手からボールがリリースされる直前のコマで一時停止します．移動ツールを選択し，ボールの上で右クリック，「軌跡追跡」を選択すると四角の枠と十字が表示されます．十字がボールの中心となるよう調整します(**図12**)．

コマ送りをしていくとKinoveaが自動でボールを検出し追尾していきます．もしボールの中心から十字がずれてきたら手動で修正します．

追跡を終わりにするときは，右クリックで「軌道の最終地点を確定」を選択します．これでボールの軌跡描画が完了です(**図13**)．

なお，ターゲットの四角上でダブルクリックすると拡大画面が出てきます．より細かく中心の位置を調整できます(**図14**)．

機能4…速度の計測

陸上の短距離では，いかに短時間でスタートからトップ・スピードまで到達し，その速度を維持できるかが好記録を出すために必要な要素となります．この

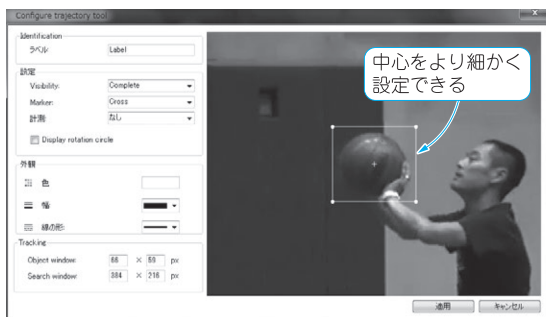


図14　ボールの中心は細かく設定可能

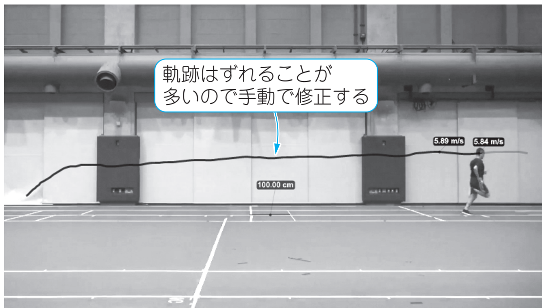


図15　スタート・ダッシュ時の頭部の軌跡

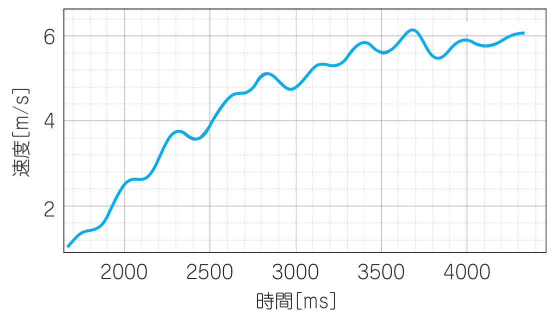


図16　水平方向の速度

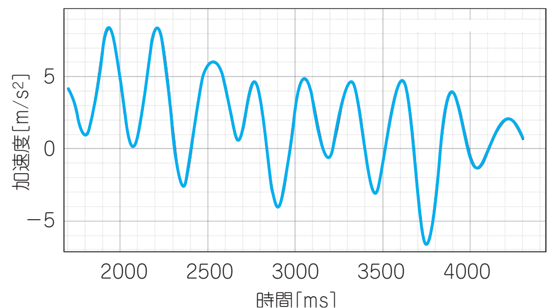


図17　水平方向の加速度

速度計測をKinoveaでは映像を利用して簡単に行うことができます．

● 垂直平面の運動の速度を計測する場合

▶撮影のコツ

- 動きと直角になる向きから撮影する
  他の計測の解説でも出てきていますが，運動方向とカメラの光軸とが直角となるようカメラの撮影位置を決めます．
- 測定したい部位にマーキング
  より正確な測定を行うために，速度計測に利用したい部位にマーキングをします．服装の色を考え，目立つ色を利用するとよいでしょう．
- できるだけ遠くからズームを使って
  カメラをパンせずに固定して撮影できるよう走る位置から離れて撮影を行います．また，レンズのゆがみの影響を減らすためにもできるだけ遠くにカメラを設置し，ズームを使ってサイズを調節します．
- 長さの基準も撮影しておく
  走る直線上に実世界での長さが分かるように，マーキングなどをしてから撮影します．

▶キャリブレーション

距離の測定同様ライン・ツールを利用して距離のキャリブレーションを行います．

▶計測

軌道追跡ツールを利用して軌跡を描きます(**図15**)．基本は自動で追尾させますが，ずれることが多いので適時手動で正しい位置に修正します．

軌跡の描画後，data analysisツールで速度だけではなく，水平成分だけの速度(**図16**)や，加速度(**図17**)のグラフなどを表示できます．

また，グラフだけではなくデータ出力も行えるので，excelなどでさらなる分析作業を行うことも可能です．

● 水平平面の運動の速度を計測する場合

距離，角度の水平平面の計測の場合と同じく，高い位置から撮影し，キャリブレーションを行います．

例えばサッカー選手の移動速度の計測など，水平平面を移動する対象の速度計測にも使えます．軌跡追跡ツールを利用して軌跡を描き，速度を算出します．高さ成分の評価はできないので，地面から浮いたボールの速度計測などには注意が必要です．

＊　＊　＊

今回はスポーツのパフォーマンスを評価するために，Kinoveaというフリー・ソフトウェアを利用して画像処理からの2次元計測を紹介しました．垂直平面か水平平面だけの計測となるため，制限の多い計測方法となりますが，手軽に行うことができるという大きなメリットがあります．手始めにKinoveaを利用してスポーツ計測を始めてみてはいかがでしょうか．

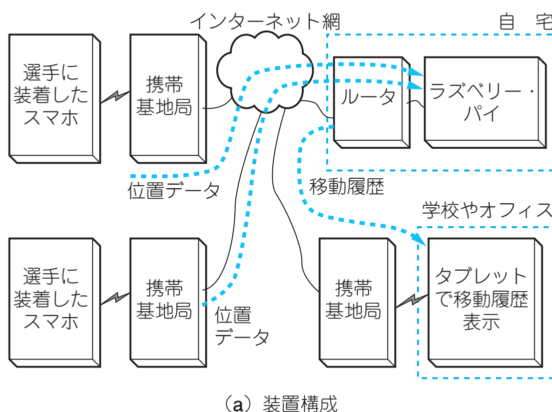
しみず・じゅん

なんてスゴイ! インターネット電子工作の世界

ラズパイ・サーバでロックオン! GPS位置トラッカ

新連載 第1回　ラズパイGPS位置トラッキング・システムの制作

村井 亮



(a) 装置構成



(b) 選手3名の移動履歴をPCで閲覧中

図1　リアルタイムに位置情報を記録しつつ閲覧も可能とするサーバをラズベリー・パイで作る

GPS (Global Positioning System；全地球測位システム) は，スマホや携帯電話，腕時計にも搭載されています．受信モジュールも2,000円から入手でき，いろいろな用途に利用できます．

本稿では，追跡対象者が装着するスマホなどのGPSデータをリアルタイムにサーバ(自宅のラズベリー・パイ)に送信し，トラッキング・データをインターネットを介してPCやタブレットに表示する，GPSトラッキング・システムの構築を行います(**図1**)．

製作したGPS位置トラッカの特徴

● 用途

▶運動選手のトレーニングに

例えば運動選手が練習中に走った道のりを監督やマネージャがモニタリングできます．選手は自身の位置を送信するGPS端末(今回はスマホまたはGPS＋ラズベリー・パイ)を身に着けて走ります．

選手の走った道のりはサーバにトラッキング・データとして蓄積されます．監督はウェブ・ブラウザからサーバへアクセスし，選手の走った経路を確認できるというわけです．各区間の記録を残してラップタイムを自動的に計算できるようにもなります．

▶大切な人の見守りに

大切な人の居場所が分からなくなっても，すぐネットワーク上で位置を確認できます．

▶自動車の追跡に

スパイ映画さながらに自動車や人を地図上でリアルタイムに追跡できます．

● サーバをラズベリー・パイで作った理由

今回，ラズベリー・パイでサーバを作ったのには次の理由があります．

▶1：有料で借りているクラウド・サーバに他人からアクセスしてほしくない

本格的な商用サーバ利用したサイトで，今回のシステムを構築するには，きちんとしたユーザ管理が必要です．特に不特定多数のユーザ管理，排他などを考慮すると，ソフトウェアが複雑になります．

ラズベリー・パイを使った自宅サーバであれば，何かあってもSDカードを差し替えれば良いので，厳格なユーザ管理が不要です．もちろん今回のシステムで

複数人のアクセス管理はできています.

▶2：無料のクラウド・サーバでは細やかなトラッキング表示やリアルタイム表示を実現できない

データ取得，単純な表示だけなら，無料サーバも使えるでしょう．しかし今回のシステムは複数人のトラッキング・データを同時に集めて，GoogleMap上へのリアルタイム表示を行っています．無料のクラウド・サーバでは，

・同時アクセス数
・送受信できるデータ数

に制限があるため，大人数から頻繁にデータを収集できません.

▶3：無料のクラウド・サーバは突然サービスを終了することもある

無料のクラウド・サーバでは，せっかく作ったシステムが突然使えなくなる可能性があります．現に2015年10月時点で無料だったFacebookのParseも2017年にサービスを終了するようです.

システム構成

図2を見てください．インターネットにつながるのはスマートフォンと制作するラズパイ・サーバです.

● 自分の位置をGPSで取得・通知するスマホなど

スマホ(GPS端末)における位置情報の取得方法は，端末の種類によって異なります．Androidであれば，LocationManagerというシステム・サービスから取得できます.

スマホ代わりにラズベリー・パイを使う場合には，USBで接続したGPSモジュールからNMEA 0183という形式のデータが定期的にシリアルで送られてくるので，これを処理して位置を取得します．GPS端末はこのように取得した位置情報をHTTP-POSTでホーム・サーバへ送信します.

HTTP通信については，Androidアプリ開発で用いるJavaでもラズベリー・パイにて使用するPythonでも完備されているので実装は容易です.

● GPS位置データを記録するラズパイ・ホーム・サーバ

今回はホーム・サーバをラズベリー・パイで作ります．ホーム・サーバは受信用のPHPのインターフェース(Apache HTTP Server上)を用いて位置情報を受け取り，これをデータベースに記録します．GPS端末は位置の変化があるたびに位置情報をサーバに送るので，サーバのデータベースには端末の移動した道のりが記録されます.

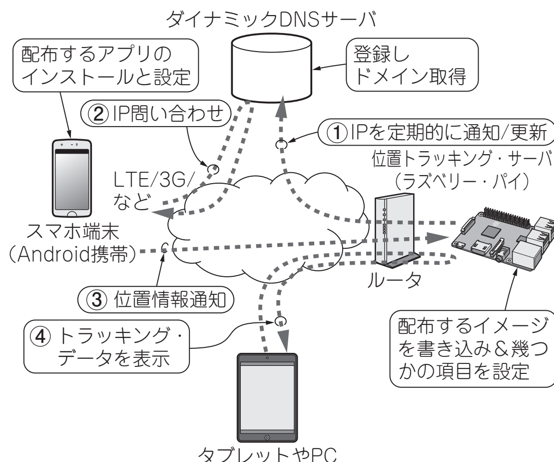


図2　GPS位置トラッカのシステム構成
スマホのGPS位置データがパソコンに表示されるまでの通信処理も併記

● 移動履歴はウェブ・ブラウザからGET

以上の過程で記録された位置の軌跡，つまりトラッキング・データは，サーバに用意したトラッキング閲覧ページ(PHP + Googleマップ)から確認できます.

移動履歴を見たい人はスマホやパソコン，タブレットからウェブ・ブラウザを用いてホーム・サーバにアクセスするだけです.

上記システムには，広く普及しているHTTPプロトコルを使い，JSON形式で時間および位置のデータを扱っているので，一般のIoT機器からの情報収集に応用できます.

必要な通信

GPSで位置データをGETしてから移動履歴がウェブ・ブラウザで閲覧できるようになるまでに必要な通信を**図2**に示します.

● ①ダイナミックDNSにIPアドレスを定期通知

ダイナミックDNSに，今回はフリーのMyDNS.JPを利用します．MyDNS.JPは自宅のホームサーバに外部からFQDN(完全修飾ドメイン名)[注1]でアクセスするためのダイナミックDNSサービスを提供します(**図3**)．ホーム・サーバは，自宅ルータのWAN-IPアドレスを，MyDNSへ定期的に登録/更新します.

● ②DNSに自宅のIPアドレスを問い合わせる

スマホはダイナミックDNSから自宅のWAN-IPを

注1：FQDN(Full Quality Domain Name)…ドメイン名とホスト名を省略せずに指定した形式．www.cqpub.co.jpなど．wwwがホスト名，cqpub.co.jpがドメイン名に当たる.

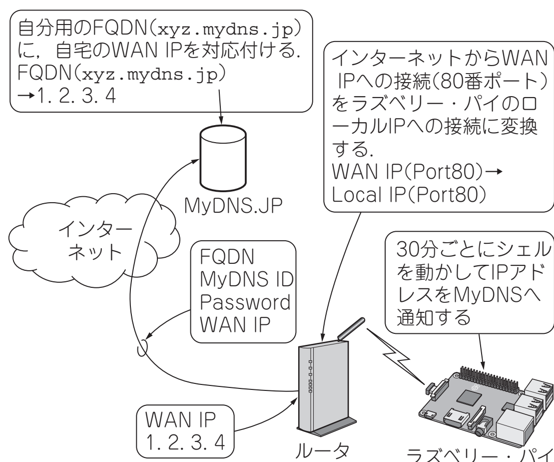


図3　フリーのダイナミックDNSであるMyDNS.JPを使って自宅のホーム・サーバに外部からドメイン名(FQDN；完全修飾ドメイン名)**でアクセスできるようにする**

取得します．これによりホーム・サーバにアクセスできるようになります．スマホの代わりにラズベリー・パイ端末を使う場合は別途，GPSモジュールとモバイル・ルータが必要です．

● ③ホーム・サーバに端末の位置情報を通知

スマホは自宅に設置したホーム・サーバに位置情報を通知します．ホーム・サーバは位置情報を受信して，保存/管理/表示を行います．なお，ルータにもポート変換の設定が必要となります．

● ④ホーム・サーバに位置データを問い合わせ

PCやタブレットではウェブ・ブラウザを利用し，ホーム・サーバに記録したトラッキング・データを表示します．

[MyDNS.JP] ユーザー登録が完了しました。
system@MyDNS.JP
送信日時:　2016/07/07 (木) 14:30
宛先:　[redacted]

[redacted]様
MyDNS.JP管理システム

ユーザー登録が完了しました

このたびはMyDNS.JPにご登録いただきましてありがとうございます。

以下に登録情報を添付しておきますので、無くさないように保管願います。

ID　パスワード

MasterID & Password :
[redacted]　[redacted]

Child ID & Password :
None

POP3/IMAP4/FTP Server = mail.mydns.jp
HTTP-BASIC URL = http://www.mydns.jp/login.html

図4　フォームで入力したメール・アドレスにはMyDNSのマスタIDとパスワードが送信されてくる

外部からラズパイ・サーバにアクセスする準備…MyDNSの登録

筆者提供のダウンロード・データをベースに，GPSトラッキング・システムの構築方法を説明します．ダウンロード・データの入手先はhttp://www.cqpub.co.jp/interface/download/contents.htmです．ダウンロードできるのは筆者提供の以下のプログラムです．

1，ホーム・サーバ(ラズベリー・パイ)のプログラム
2，スマートフォンのアプリ

● ラズパイに★★★.mydns.jpでアクセスできる

今回は自宅のラズベリー・パイをホーム・サーバにするので，自宅のWAN-IPにFQDNを割り当てる必要があります．MyDNS.JP(以下MyDNS)は，フリーで利用できるダイナミックDNSサービスです．今回はGPS端末や外部のウェブ・ブラウザからホーム・サーバに，このダイナミックDNSを利用して，自宅にFQDNの名前でアクセスします．こうすることで，自宅のインターネット環境のようにWAN IPが変わっても，常にFQDNでアクセスできるようになります．

● 利用者登録

まずはMyDNSの登録を行い，ホーム・サーバからWAN IPの通知を受け取ることができる状態まで設定を行いましょう．

MyDNSのトップ・メニューから「JOIN US」へ行き，個人情報を含むフォームを入力します．入力を終えると確認画面が表示され，[OK]を押すことで登録が完了します．

フォームで入力したメール・アドレスには，MyDNSのマスタIDとパスワードが送信されてきます(**図4**)．このIDとパスワードは後でホーム・サーバからWAN IPアドレスの更新を行う際のBasic認証で必要となるのでメモを取っておきます．

● 独自ドメイン取得

メールで送信されたIDとパスワードをトップ・ページの「User Login」に入力してアカウントにログインします．次に「Welcome Administrator」のサブメニューより「DOMAIN INFO」へ飛び，**図5**のフォームを入力して独自のドメインを取得します．

最後に同じサブメニューから「IP ADDR DIRECT」(**図6**)へ飛び，動的なDNS設定となっていることを確認します．

このIDには，一つのFQDNが対応します．このた

め，以後，FQDNの更新には，IDに対応したWAN IPを通知するだけで，FQDNは使わないことを気をつけてください．

ラズパイ位置トラッキング・サーバを作る

● 位置トラッキング・サーバのイメージをラズパイ用SDカードに書き込む

今回作成したシステムで利用できるラズベリー・パイのイメージを本誌ウェブ・ページからダウンロードできます．ダウンロード・イメージをSDカードに書き込んで手持ちのラズベリー・パイに入れれば，一通りシステムのホーム・サーバに必要なソフトウェアを利用できます．

イメージ(`raspberry-pi-gps-tracking-server.img`)の容量は8Gバイトなので，余裕をもって16GバイトのSDカードを使用するとよいでしょう．8GバイトちょうどのSDカードの場合には微妙な容量の違いによって書き込むことができないことがあります．また，SDカードへのイメージの焼き方は公式サイトの該当項目，

```
https://www.raspberrypi.org/
documentation/installation/
installing-images/README.md
```

を参照してください．**表1**に配布するイメージのユーザ情報を示します．

● 書き込んだ後にやること

SDカードにイメージを書き込んだだけでは外部からサーバとなるラズベリー・パイにアクセスできません．以下を実行します．

ステップ1：ローカルIPアドレスの固定
ステップ2：外部からのアクセスを許可するためのルータの設定
ステップ3：MyDNSへWAN IPアドレス通知を行うためにサーバ内mydns_poke.shの書き換え



図6　「IP ADDR DIRECT」へ飛び，動的なDNS設定となっていることを確認

設定できるTXTレコードは一レコードあたり**250**文字までとします。

例:

Hostname*	Type*	Content
_spf	TXT	v=spf1 +ip4:210.197.72.31 +a:mail1.big.or.jp
google._domainkey	TXT	v=DKIM1; k=rsa; t=y; p=apcfnSVG5YWaSoBTSH84hSBB84B4322h...

注意！**TXT**レコードの形式はチェックしませんのでその設定には十二分にご注意下さい。

※子**ID**はドメイン名を変更することはできませんのであらかじめご了承下さい。

*印は必須項目です。

Domain* : (FQDN)
gps-tracking.mydns.jp
MX : (Hostname, Priority. FQDN)
10
10
10
10
10
10
10

これが自宅ルータまでのURLとなる

図5　フォームを入力して独自のドメインを取得

なお，サーバをインターネットに公開した状態にするので，本GPSトラッキング・システムを試用してみる際には，アカウントも知れ渡っている`pi`以外のユーザに変えるべきです．最低限スーパユーザと`pi`のパスワードは変更してください．これを行わないとラズベリー・パイが乗っ取られて，最悪，犯罪に使われる恐れもあります．

● ステップ1…IPアドレスの固定

ステップ2のルータの設定で，ルータ側にポート変換またはDMZの設定を行う際に，ホーム・サーバのローカルIPアドレスを固定しておく必要があります．起動したラズベリー・パイでLAN内のIPアドレスの固定作業を行いましょう．ただし，他の端末とIPアドレスがぶつかると通信ができなくなりますので，IPを固定する際は，ご使用中のルータでDHCPリースの状況をよく確認してください．

▶ (1) `/etc/dhcpd.conf`による設定

ホーム・サーバ側の設定で固定する場合，`/etc/dhcpcd.conf`のファイル末尾に，**リスト1**に示す設定を加えます．ただ，この方法は今までもRaspbianのバージョンが変わると変更されています．

▶ (2) GUI画面による設定

Raspbian起動後のウィンドウ画面において，右上のネットワーク・アイコンを右クリックすると，**図7**の画面が出ます．ここで有線LANやWi-Fi無線LANのIPアドレスを設定することもできます．この方法

表1　配布サーバ・イメージのユーザ情報

項　目	入力内容
ユーザ	pi
パスワード	raspberry
MySQLユーザ	root
MySQLパスワード	raspberry

リスト1　IPアドレスの固定…/etc/dhcpd.confによる設定

```
interface eth0
static routers=192.168.0.1           ← ルータのIP
static domain_name_servers=
static ip_address=192.168.0.127      ← 固定したいIP
static domain_search=
```

(a) イーサネットの場合

```
interface wlan0
static routers=192.168.0.1           ← ルータのIP
static domain_name_servers=
static ip_address=192.168.0.128      ← 固定したいIP
static domain_search=
```

(b) 無線LANの場合

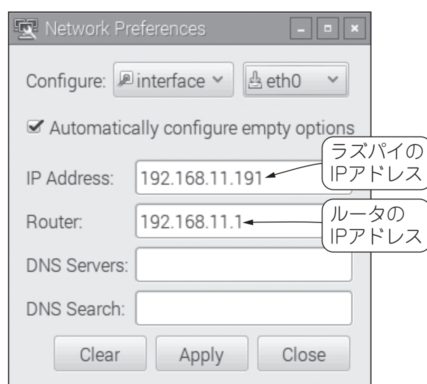


図7　ラズパイ上でIPアドレスを固定

もRaspbianのバージョンによって，GUIの場所がよく変更されますので，どちらが良いかは一概に言えません．

▶(3) ホーム・ルータによる設定

本来，ラズベリー・パイのLocal IPアドレスは，デフォルトのホーム・ルータからのDHCPによる自動付与のままにして，ホーム・ルータ側でIPアドレスを固定する方法が，アドレスの衝突が起きないので一番安全です．

ラズベリー・パイで，次のようにコマンドを打って，ラズベリー・パイのMACアドレスを確認します．

```
$ ifconfig
eth0        Link  encap:Ethernet
HWaddr b8:27:eb:74:a1:3e  ← 有線LANのMACアドレス
中略
wlan0       Link  encap:Ethernet
HWaddr b8:27:eb:21:f4:6b  ← 無線LANのMACアドレス
```

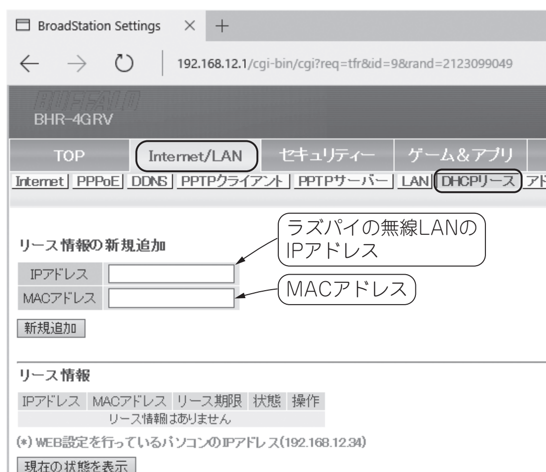


図8　ルータでラズパイのIPアドレスを固定

後はホーム・ルータのDHCPの設定で，このMACアドレスに割り当てるローカルIPアドレスを設定します．ルータの設定方法は機種によってざまざまです．BHR4GRV（バッファロー）の例を示すと，Internet/LANの中に，DHCPリースがあります（**図8**）．ここでラズベリー・パイのMACアドレスにIPを割り当てることができます．

● ステップ2…ホーム・ルータの設定

今回，システムの要件からホーム・サーバ宛てのHTTPリクエストの許可が必要となります．しかし一般には，ホーム・ルータは許可されていない外部からのアクセスをファイア・ウォールで防ぐ役割を果たします．そこでルータの設定を行い，固定したホーム・サーバのLAN IPに対しては，HTTPリクエストを許可するようにします．通常ホーム・ルータでは，次の二つの方法のうち，どちらかは利用可能ですので，どちらか一方を設定してください．

▶第1の方法…ポート変換の設定

第1の方法はポート変換によるものです．使っているホーム・ルータによって，アドレス変換，ポート・フォワードという言葉が使われていますので，ご利用中のホーム・ルータのマニュアルをよく見てください．

図9に示すように，「ポート変換」機能の設定で，インターネットからホーム・ルータに送られてきたHTTPリクエストをホーム・サーバへ転送するように設定します．

転送先は先ほど固定したホーム・サーバのLAN IPアドレス（例えば192.168.0.127や192.168.0.128）です．これによって以後TCP80番（HTTP）への要求はホーム・サーバに対して振り分けられるようになります．

この機能は，ゲームでよく使う機能のため，ホーム・ルータでも，**図9**のように「ゲーム＆アプリ」の中にこの設定項目が存在することがあります．また，ポート変換という名前も，アドレス変換，ポート・

フォワードなどの名前の設定項目となっている場合も多いので，よくマニュアルを確認してください．

▶第2の方法…DMZの設定

第2の方法はDMZを利用することです．**図10**に示すように，「DMZ」(この項目は使用するルータによって異なる)にホーム・サーバのIPアドレスを入力します．DMZにホーム・サーバを指定すれば，アプリケーションで必要な通信はまず容認されますので，外部からのアクセスが可能となります．

ただし，DMZは外部の要求について全面的に指定されたLAN IPへ送信を割り当てるため，セキュリティ的に安全ではありません．こちらを利用するときは注意が必要となります．

DMZは，もともとは，DeMilitarized Zone(非武装地帯)の意味ですが，このようにセキュリティ上外部から直接アクセスできる場所のことをいうようになりました．

● ステップ3…MyDNSへのWAN IPアドレス通知

ホーム・サーバのWAN IPアドレスはいつ変わってしまうか分かりません．そこでMyDNSに登録されているIPアドレスは定期的に通知をして，更新しなくてはなりません．今回はラズベリー・パイからMyDNSへ定期的にIPアドレスを通知することで，この更新を行います．通知にはHTTP-BASICによる認証を使用します．

MyDNSではIPアドレス通知用URLというものが用意されています．このURLにはwgetでアクセスできますから，後はLinuxの自動実行サービスcronで一定時間おきにこの通知を行えばよいということになります．

今回は**リスト2**のようなスクリプトを用意しました．スクリプトには変数としてMyDNSのユーザIDとパスワードを持ちます．これらをwgetでBASIC認証する際の引き数として与え，IPアドレス通知用URLにアクセスします．これでIPアドレスが更新されます．

このIP通知スクリプトは，ダウンロードしたサーバ・イメージでは，/home/pi/の下に入っています．MyDNSの登録を行ったときにメモしたIDとパスワードを適宜，サーバのmydns_poke.shの変数部分に代入して使ってください．このコマンドを実行するとIPアドレスの通知が行われます．

この通知作業を自動化するためにはcronに登録しなくてはなりません．今回は毎時0分と30分の2回，MyDNSに通知を行います．cronの設定ファイルを編集するにはcrontabのコマンドで編集オプションをつけて実行します．なお，crontabの実行初回は設定ファイルを編集するエディタを選ぶことになります．

```
crontab -e
```

cron設定ファイルの最後に次のような一文を加えましょう．分(m)，時(h)，日(dom)，月(mon)，曜日(dow)の形式で自動実行する時間を指定し，その後ろに実行したいコマンド(command)を指定しています．今回は用意したスクリプトを実行するのでコマンドはbashでスクリプトを実行する形になっています．

```
# m h  dom mon dow   command
0,30 * * * * /bin/bash /home/pi/
mydns_poke.sh
```

それぞれの時間についてはワイルドカード(＊)や時間の範囲を表すハイフン(3-5)，カンマで列挙し

TOP | Internet/LAN | セキュリティー | ゲーム&アプリ
ポート変換 | DMZ | UPnP | QoS

ポート変換の 新規追加

グループ	新規追加 新規追加:
Internet側IPアドレス	ブロードステーションのInternet側IPアドレス 手動設定:
プロトコル	○全て ○ICMP ○任意 プロトコル番号: ◉TCP/UDP 任意のTCPポート 指定の仕方 任意のTCP/UDPポート:
LAN側IPアドレス	192.168.12.34
LAN側ポート	TCP/UDPポート:

固定したラズパイのIPアドレス

図9　ルータのポート変換の設定
外部からポート80へのアクセスは通過させる

TOP | Internet/LAN | セキュリティー | ゲーム&アプリ
ポート変換 | DMZ | UPnP | QoS

DMZのアドレス
※WEB設定を行っているパソコンのIPアドレス[192.168.12.34]
設定

ラズパイのIPアドレスを入力

図10　ルータのDMZの設定
インターネット側からのアクセスが素通しになるので詳しい人向け

リスト2　mydns_poke.sh
使用するときにはwgetの直前の#を抜く

```
#!/bin/bash

user="your-user"     ← 取得したMyDNSのユーザID
pass="your-pass"     ← 取得したMyDNSのパスワード

#wget --http-user=${user} --http-passwd=${pass}
-O - http://www.mydns.jp/login.html
↑ 使用するときに消す
```

表2　cronで時刻に設定可能な値

フィールド	指定可能範囲
minute（分）	0～59
hour（時）	0～23
day of month（日）	1～31
month（月）	1～12または月名（Janやfeb）
day of week	0～7（0と7は日曜日）または曜日名（Sunやmon）

た値（0,30），範囲と実行間隔とを使用した表記（*/10）などの指定方法があります．**表2**にcronで時刻に設定可能な値を示します．**表3**にcronに設定する時間の例を示します．

なお，今回配布しているラズベリー・パイ・サーバでは，既に/home/pi/mydns_poke.shについてcronの設定が施されています．そこで，使用する場合だけに限ればmydns_poke.shへ取得したMyDNSアカウントのユーザIDとパスワードを書き込んでコメント化されている，wgetの最初の#を削除して生かしてください．

● 早速試してみよう

ここまでの設定が終了したら，MyDNSで取得したFQDNを用いて次のアドレスにアクセスしてみましょう．

```
http://[Raspberry Piサーバ のFQDN]/gps-tracking/menu.php
```

うまく設定できていれば，MyDNSへホーム・ルータのWAN IPはMyDNSへ通知され，外部からFQDNを使ってHTML接続できます（**図11**）．

自宅のネットの中で接続するときは，ホーム・サーバに設定したローカルIPを直に入れてください．

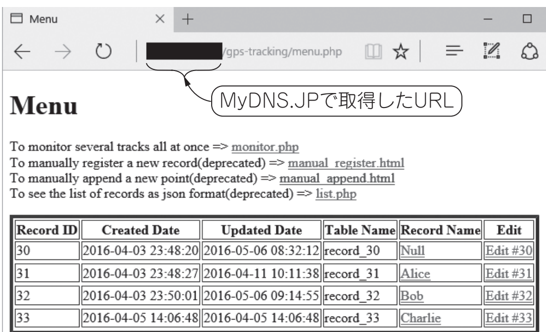


図11　うまく設定できていればMyDNSへホームルータのWAN IPはMyDNSへ通知され外部からFQDNを使ってHTML接続できる

表3　cronに設定する時間の例

例	意　味
*	ワイルド・カード
2,5	2分と5分，火曜と金曜，2月と5月など
3-8	3，4，5，6，7，8に同じ
0-23/2	0，2，4，6，8，10，12，14，16，18，20，22に同じ
*/10	10分ごと，10時間ごとなど

GPS端末（Androidスマホ）の設定

システムで使用するスマートフォンのGPSトラッキング・アプリをダウンロードできます．アプリケーションのパッケージは，app-release.apkで終わる三つの種類を用意しました（**表4**）．手持ちのスマホ（Andoird端末）にインストールすることで，配布のラズベリー・パイ・サーバ・イメージと連携して動作するGPSトラッキング・システムを利用できます．

なお，実際のシステムにおける利用の際にはGPS Trackingというアプリを使ってください．Easy GPS Trackingは，次回以降，プログラムを説明するための簡略化したバージョンです．今回はGPSの位置追跡が最も容易にできるGPS Trackingのアプリを例に，アプリの設定方法を説明します．

● アプリのダウンロード

【Downloads】にあるStandardGpsTracking-app-release.apk，GpsTracking-app-release.apkなどは，それぞれ**表4**のアプリに対応したパッケージになっています．

保存はPC経由でUSB接続にて行うのがよいでしょう．マウントをすれば，一般のファイル・マネージャ・アプリからタップ&インストールできます．

今回は簡単のため配布のイメージから構築したラズベリー・パイ・サーバからAndroid用のGPSトラッキング・アプリをダウンロードする方法を示します．今回配布しているサーバのイメージでは/var/www/html/android-appにAndroidアプリのapkファイル（アプリケーション・パッケージ・ファイル）を置いています．

表4　アプリケーションの対応

apkファイル	アプリ	補　足
EasyGpsTracking-app-release.apk	Easy GPS Tracking	アプリの解説のための最小版
StandardGpsTracking-app-release.apk	Standard GPS Tracking	UIを整えた最小トラッキング・アプリ
GpsTracking-app-release.apk	GPS Tracking	多数機能を追加したトラッキング・アプリ

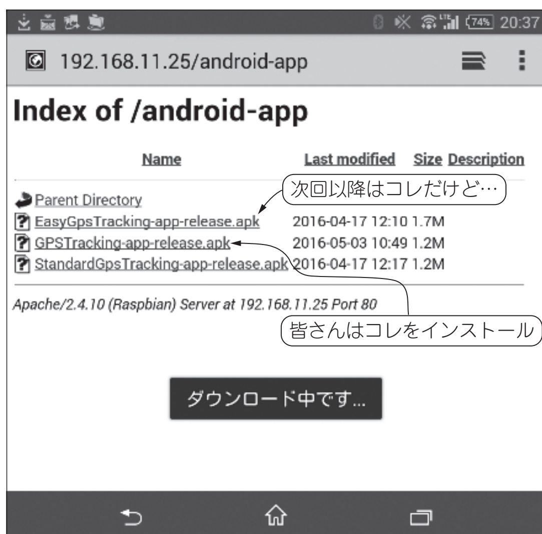


図12　筆者提供のアプリケーション・パッケージ・ファイルをダウンロード

端末のブラウザから3タップ(ダウンロード，apkファイル選択，インストール同意)で簡単にインストールできます．まず**図12**に従ってアプリケーション・パッケージ・ファイルをダウンロードしましょう．アクセスすべきURLは次の通りです．

```
http://[Raspberry Piサーバ のFQDN]/
android-app
```

ブラウザからダウンロードしてきたファイルは

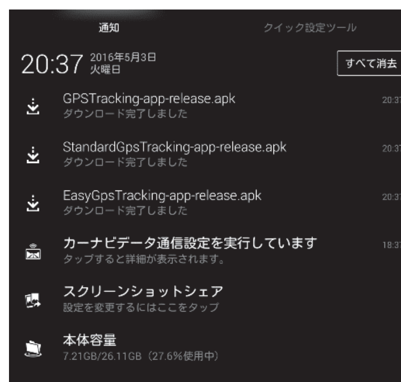


図13　ブラウザからダウンロードしてきたファイルは通知欄に一覧表示される

図13のように通知欄に一覧表示されています．ラズベリー・パイ・サーバからダウンロードしたこれらのapkファイルをタップすると，**図14**のようにインストール許可が求められるので，インストールしてください．なおこの際，**図15**のように，あらかじめ設定から提供元不明のアプリの導入を許可しておく必要があります．

● GPSトラッキング・アプリの設定

表4に示したように，端末アプリケーションとして実装の極めて簡潔なEasy GPS Trackingから一般的な実用レベルに足るトラッキング・アプリであるGPS Trackingまで三つをダウンロードできるようになっています．ここではその中でもGPSの位置追跡が最も容易にできるGPS Trackingのアプリ(**図16**)を例に，アプリの設定方法を説明します．

▶位置情報の保存テーブル作成

まずトラッキングを行うためにはサーバのデータ・ベースに位置情報を保存するためのテーブルを作成する必要があります．このレコードをサーバに新規作成するには二つの方法があります．一つはアプリ側のRecord Manipulation→Register Record→「REGISTER」から新規のレコードの名前を決めて登録・作成する方法です(【レコード新規登録(Androidアプリ)】を参照)．

もう一つの方法はウェブ・ブラウザからサーバの次



図14　筆者提供のアプリのインストールを許可

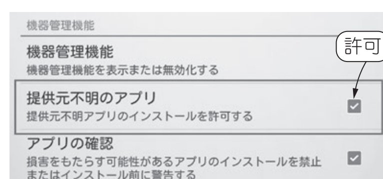


図15　Androidアプリ提供元不明を許可

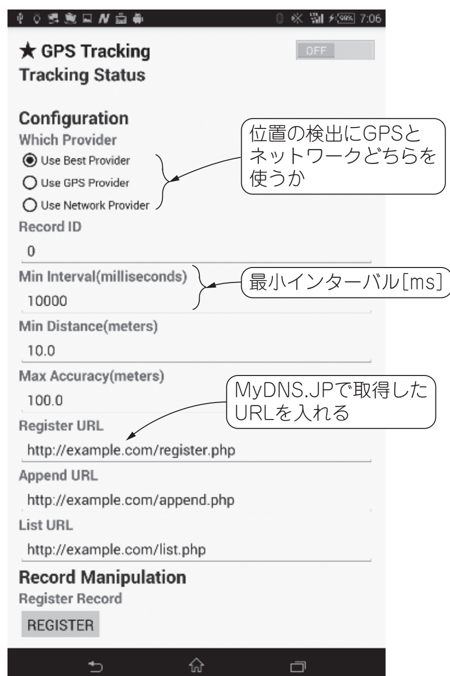


図16　スマホにインストールしたアプリGPS Trackingの外観

のURLにアクセスして，レコードの名前をフォーム に入力した上で作成する方法です(**図17**).

```
http://[Raspberry Piサーバ のFQDN]/
gps-tracking/manual_register.html
```

▶位置情報はGPSを使うかネットワークを使うか

続いてアプリケーションのConfigurationの項(**図16**上)にはトラッキングをするにあたって必要な設定が入力されています．画面上部にあるラジオ・ボ

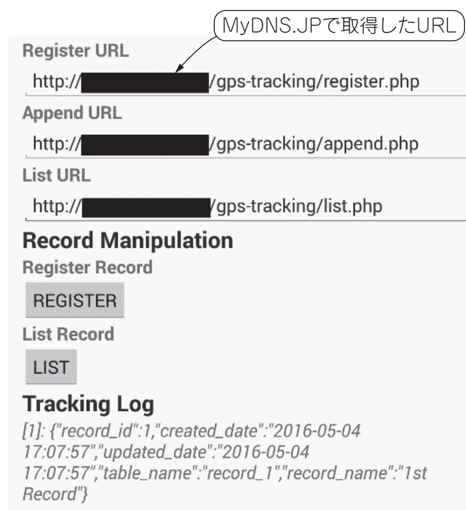


図18　ラズパイ・サーバへ接続する際のURLを三つ登録する

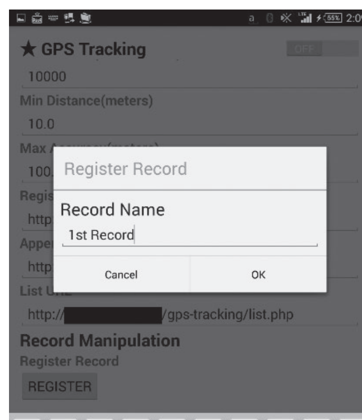


図17　レコード新規登録(Androidアプリ)
レコードを登録するとアプリ内Configurationにあるパラメータ「Record ID」が新規作成したレコードのものに自動変更される

タンは，AndroidのGPS取得に際して，端末本体のGPSで手に入れた位置情報を使うのか，ネットワークから得たおおよその位置情報を使うのか，それともその場に応じた最適の手段で位置情報を取得するのかを設定します．

「Record ID」とは，サーバに既に作成されて登録されているどのレコードに対して位置情報を送るのかを決める整数のパラメータです．例えばRecord IDが30，31，32，33などとサーバに登録されている場合，GPS端末はこれら四つのレコードのどれかに向けて位置情報を送信することになります．「Record ID」が例えば32だとすると，GPS端末から送信された位置情報はサーバの32番レコードに順次蓄えられていきます．

▶位置の取得頻度

GPS端末では幾つかのパラメータで取得頻度の制限をしています．つまり，「Min Interval(GPS位置情報取得の最小時間間隔)」，「Min Distance(GPS位置情報取得の最小距離)」，「Max Accuracy(サーバに送信してもよい位置情報の不正確さの最大値)」がそれです．

位置の取得頻度が極端に高かったり低かったりするのはトラッキングとしての見栄えが悪くなるので，デフォルトでは10秒経過，または10mの移動につき位置情報の取得とサーバへの送信を行う設定となっています．位置の精度については取得した経度緯度から半径何mに何%の確率でいるのかという情報なのですが，概して50～100mで十分だと思います．

▶サーバへの登録情報

最後の設定としてはGPS Trackingの場合にはサーバのPHPインターフェースのURLを三つ登録する必要があります(**図18**)．それぞれ`register.php`は

リスト3　筆者提供のAndroidアプリに設定すべきURL

```
http://[Raspberry Piサーバの FQDN]/gps-tracking/register.php   ← Register URL
http://[Raspberry Piサーバの FQDN]/gps-tracking/append.php     ← Append URL
http://[Raspberry Piサーバの FQDN]/gps-tracking/list.php       ← List URL
```

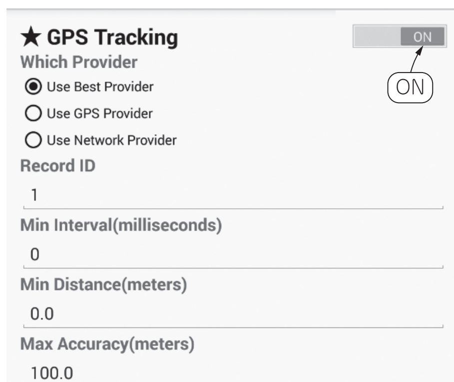


図19　トラッキング開始

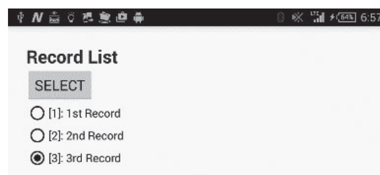


図20　レコードをリストから変更する

レコードの登録，append.phpは位置情報の受信，list.phpは現在サーバに保存されているレコードのリストのJSON出力を行うインターフェースとなっています．ここまでのシステムの構築を厳密に行っている場合，これらのファイルのURLは**リスト3**の通りとなります．

● GPSトラッキング・アプリの使用

GPS Trackingのアプリの設定が終了したら，いよいよ位置の追跡を行ってみましょう．アプリの右上にあるスイッチをOFFからONに切り替えることでトラッキングがスタートします(**図19**)．

▶GPS機能は電力食い

スマートフォンのような端末の場合GPS機能というのは常時ONにしておくと電池を猛烈に消費していく悪夢の機能だったりします．いつもはGPSの設定を切っているという方はこの機会にONに，いつもONにしているという方は電池残量に気を付けてお出かけください．

トラッキングを終了してサーバへの位置の送信を終了するときにはアプリの右上部のスイッチを今度はOFFに切り替えてください．サービスとして半常駐するGPSトラッキングの処理が終了します．

● GPSトラッキング・アプリの機能

GPSのトラッキングだけが行えるアプリケーションとして開発していた今回のGPS端末ですが，ユーザビリティを考えて位置情報を送信するレコードについては，UIから直観的に変更できる工夫が加えてあります．サーバ側にレコードが一つ以上ある状態で，Record Manipulation→Register Record→「LIST」と進んでください．URLにて設定したlist.phpの情報をもとにラジオ・ボタンのレコード選択画面にアプリが遷移します(**図20**)．ここからトラッキングを記録するレコードを選べば，比較的にスムーズにシステムの実験が行えます．

完成！…GPS位置トラッキング・データを地図で表示する

保存したトラッキング・データには，サーバの/gps-tracking/menu.php上のメニュー欄に表示されたレコードのリンクに飛ぶことでアクセスできます．トラッキング・データはリンク先のtrack.phpにてGoogleマップ上にプロットされ，同時に一覧としてHTMLの下部に列挙されます．

また，/gps-tracking/monitor.phpでは複数のレコードを色分けして一度に表示できます．monitor.phpのページ下部にある表の「Track Color」へHTMLで使える色(例えばredや#00ffffなど)を指定することでトラックが表示されます．マラソン・ランナの団体をコーチングする場合，このサーバにランナの位置情報を集約させれば，誰がどこにいるのか一目瞭然でしょう[**図1**(**b**)]．

＊　　＊

移動トラッキング・サーバのインストール方法を紹介しました．ラズパイ・サーバの設計方法やAndroidアプリの設計方法については，次号以降のラズベリー・パイ・コーナで紹介します．さらに，位置をサーバに通知するスマートフォンを「ラズベリー・パイとGPSモジュール」に置き換える方法も同コーナで紹介します．

むらい・りょう

8000円で試せるFPGAアシスト制御の世界

はじめてのARM Cortex-M3 ×FPGAマイコン

新連載 第1回 **約8000円! Cortex-M3コア内蔵FPGA SmartFusion2入門キット**

浅井 剛

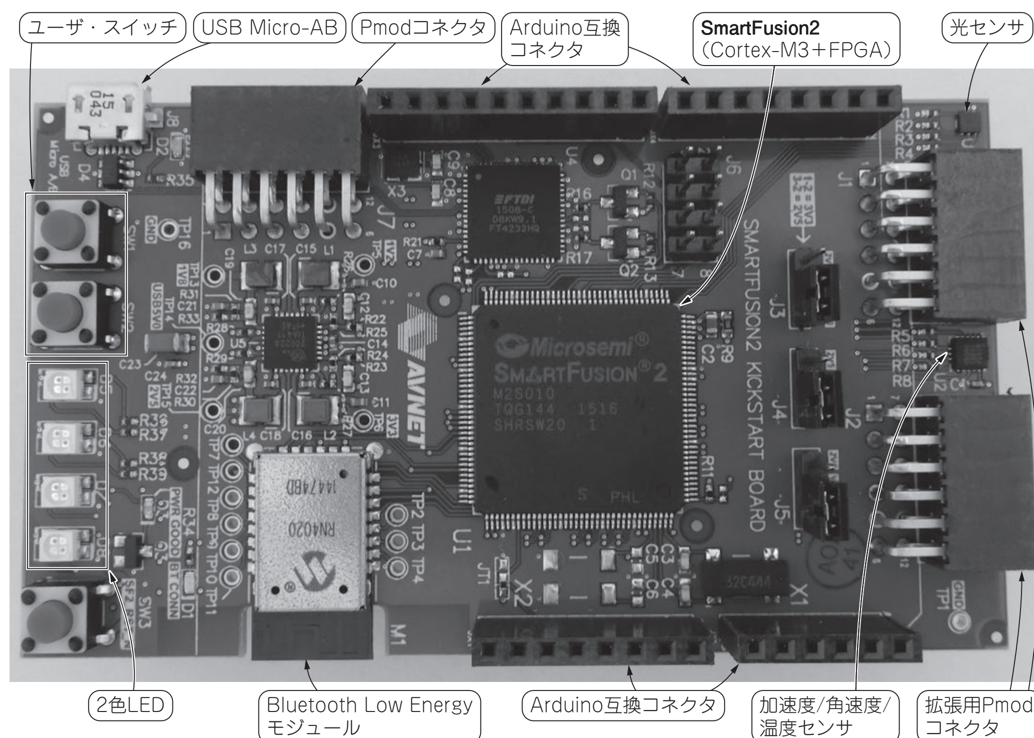


写真1 Cortex-Aだけじゃなかった! 8,100円のARM Cortex-M3コア内蔵FPGAの入門キットSmartFusion2 KickStart Kit

このコーナでは，ARMプロセッサとFPGA (Field Programmable Gate Array) が1チップになったザイリンクスのZynqと，アルテラのSoC (Cyclone SoCやStratix SoC) を対象に，うまく使う方法やさまざまな話題を取り上げてきました．これらは，Cortex-AプロセッサでLinuxが動作するパワフルなデバイスです．マイコンが使われているような用途では過剰性能になりがちです．

今回は，ARMプロセッサとしてCortex-Mを内蔵し，制御用途で使いやすいFPGA「SmartFusion2」を搭載した低価格キットを取り上げます．（編集部）

ARMプロセッサがハード・マクロで搭載されているFPGA (Field Programmable Gate Array) といえば，ザイリンクスやアルテラ (インテルの1部門) の製品を思い浮かべる方が多いと思います．Microsemi (旧アクテル) にも，Cortex-M3プロセッサを搭載したSmart Fusion2があります．

今回はSmartFusion2を搭載した開発キットSmart Fusion2 KickStart Kit (アヴネット，**写真1**) を使ってみます．

特徴

SmartFusion2 KickStart Kit (以下，キット) の機能と構成を**表1**と**図1**に示します．

表1　ARM Cortex-M3コア内蔵FPGA入門キットSmartFuion2 KickStart Kitの仕様

項　目	仕　様	型名・詳細
ARM FPGAデバイス	SmartFusion2	M2S010TQG144（Microsemi）．Cortex-M3＋約1.2万LUT規模FPGA
動作クロック	水晶発振器	50MHz
通信機能	Bluetooth 4.1 Low Energy	RN4020-V/RM（マイクロチップ・テクノロジー）
	USB-UART変換	FT4232H（FTDI）
センサ	光センサ	MAX44009（マキシム）
	加速度/角速度/温度センサ	MAX21105（マキシム）
ユーザ入出力	2色LED×4個	赤/緑/黄（赤＋緑）
	スイッチ×2個	プッシュ
拡張インターフェース	Pmodコネクタ×3	12ピン・タイプ
	Arduino Shieldコネクタ	3.3Vのみ対応
電源	USB給電（5V）	MAX20022（マキシム）

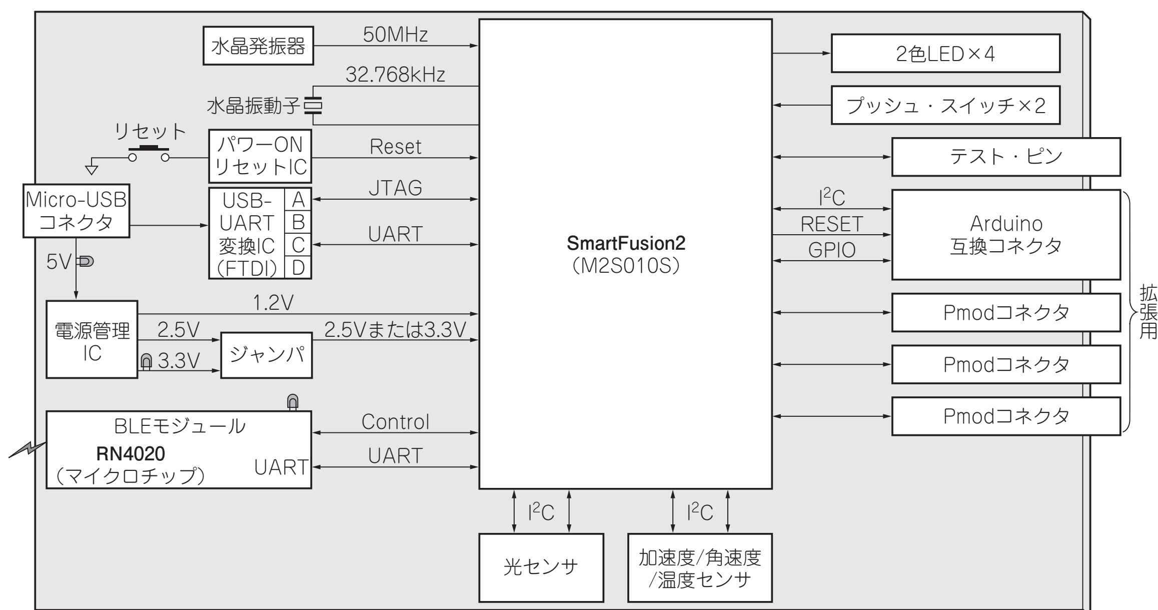


図1[1]　ARM Cortex-M3コア内蔵FPGA入門キットSmartFusion2 KickStart Kitの回路構成

● ARM FPGA内蔵機能をそのまま試せるシンプル構成

ボード上の入出力デバイスやコネクタが，SmartFusion2とほぼ直接接続しているという簡単な構成です．

外部メモリは搭載されていません．Cortex-M3プロセッサ用のプログラムやワーク・エリアは，SmartFusion2に内蔵されているメモリのみを使用して動作させます．内蔵メモリは，256Kバイトの不揮発性メモリ（eNVM），64Kバイト（誤り訂正未使用時は80Kバイト）のRAM（eSRAM）のみです．

● 拡張インターフェース…定番Arduinoシールド＆Pmod対応

機能拡張用には，最近多くの評価キットに搭載されているArduinoシールド・コネクタと，3個のPmodコネクタが用意されています．

Pmodは，Digilent社が仕様を策定した拡張インターフェースです．UART，I^2C，SPIといったシリアル通信を備えた周辺モジュールや，GPIOによる接続が容易になります．Digilent社とマキシム（Maxim Integrated Products社）が多くのモジュールを提供しています．

キットにはArduinoシールドと2個のPmodコネクタに特定のボードを想定したデモ回路/プログラムがあらかじめ書き込まれています．後述のテスト・ユー



図2　キットを接続しているCOMポートは自動検索される

ティリティ・プログラムもそれを前提に作成されています．

● 惜しい！ ちょっと残念な点…高速インターフェースを持たない

本キットは，8,100円と魅力的な価格の反面，イーサネット，USB，CAN，PCI Expressなどの高速シリアル通信はサポートしていません．キットに搭載されているSmartFusion2デバイスには内蔵されているので，汎用でかつ手元の環境で評価しやすいイーサネットかUSBのどちらか一方はサポートしてほしかったと思います．

使ってみる

● ステップ1：準備

出荷状態のキットを動かす前に，テスト・ユーティリティ・プログラム（`KickStart-Windows-Test-Utility.zip`）と，USB-UART/JTAG変換LSI（FT4232H）のドライバ・ソフトウェア（`USB_Driver.zip`）を，アヴネットのサイトからダウンロードし，ホストPCにセットアップしておく必要があります．ダウンロードにはアブネットのサイトへの登録（無償）が必要です．

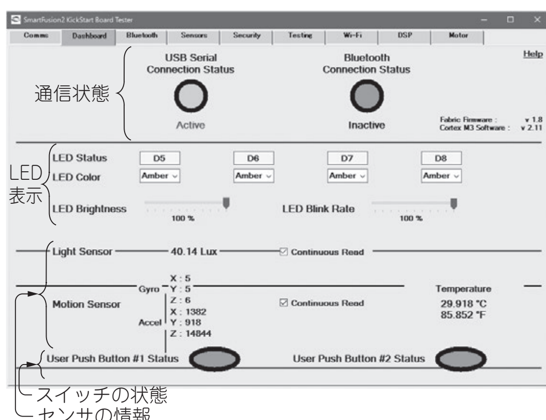


図3　GUIからボード上の各デバイスの入出力テストが行えるようになっている

● ステップ2：プログラムの設定

ホストPCとキットをUSBケーブルで接続してからテスト・ユーティリティ・プログラムを起動すると，**図2**に示すダイアログが表示されます．

USB-シリアル変換デバイスがどのCOMポートに割り当てられるかは使用するUSBポートなどの状況で異なります．このため，多く開発ツールではWindowsのデバイス・マネージャで割り当てられたCOMポート番号を事前に確認しておく必要があります．

しかしこのテスト・ユーティリティ・プログラムは，起動すると活性化されているCOMポートをスキャンし，キットに接続されているポートを自動で設定してくれます．この機能は大変便利です．ぜひ他社のソフトウェアでも参考にしてほしいと思います．

● ステップ3：テスト・ユーティリティの操作

ダイアログ中COMポート情報の下にある「Open Port」をクリックすれば，テスト・ユーティリティを利用できるようになり，**図3**の画面（Dashboard）が表示されます．

最上部には，キット外部との通信インターフェースであるUSBシリアルとBluetoothの接続状況が丸形の色と文字で表示されます．

その下段はドットLEDの点灯制御です．リスト・ボックスで，個別に消灯（Off）/赤（Red）/緑（Green）/黄（Amber）を選択できます．また4個共通ですが輝度（Brightness）と点滅間隔（Blink Rate）を0～100%で設定できます．

ドットLEDの下は，光センサと，ジャイロ（角速度）/加速度/温度センサからの情報です．光センサの情報は照度（Lux），温度の情報は摂氏（℃）と華氏（°F）に換算されて表示されます．角速度/加速度の情報はそのまま表示されているようです．「Continuous Read」にチェックを入れると，リアルタイムなデータを表示することができます．

最下段にはプッシュ・スイッチの押下状態が表示されます．

● ステップ4：ホストPCとの通信

各種データを一括表示するDashboard以外に，ホストPCとの通信状況をメインに表示するComms画面（**図4**）や，機能ごとにより詳細な情報を表示する画面（**図5**）が用意されています．Comms画面はキットとホストPC間の通信状況が生で見られるため，デバッグ用と思われますが，拡張ボードをカスタマイズした際にも活用できそうです．



図4　ホストPCとの通信状況を表示
右側に全体の状態とUSB-シリアルの送受信状況が見える

● スタータ・キットならではは制約が残念

このツールには，特定のPmodモジュールやArduinoシールドを使うことを前提としたテストも含まれています．しかしキットには付属していませんので，筆者は試してみることができませんでした．

キットなのに，テスト・プログラムで使えない機能が多いのは，改善の余地があると思います．

また，発表から1年程度しか経過していないためか，テクニカル・ブリーフやアプリケーション・ノートなど，このキットを活用する上で必要な情報が十分とは言えない印象です．今後有益な情報が充実されていくことを期待したいと思います．

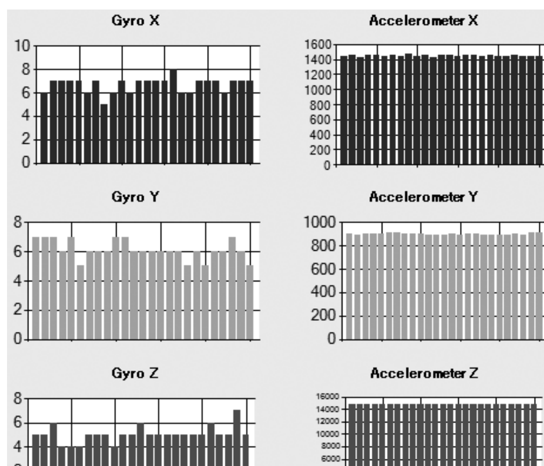


図5　機能ごとの詳細表示
"Continus", " Show Graphs" と" AutoScale" にチェックを入れた状態

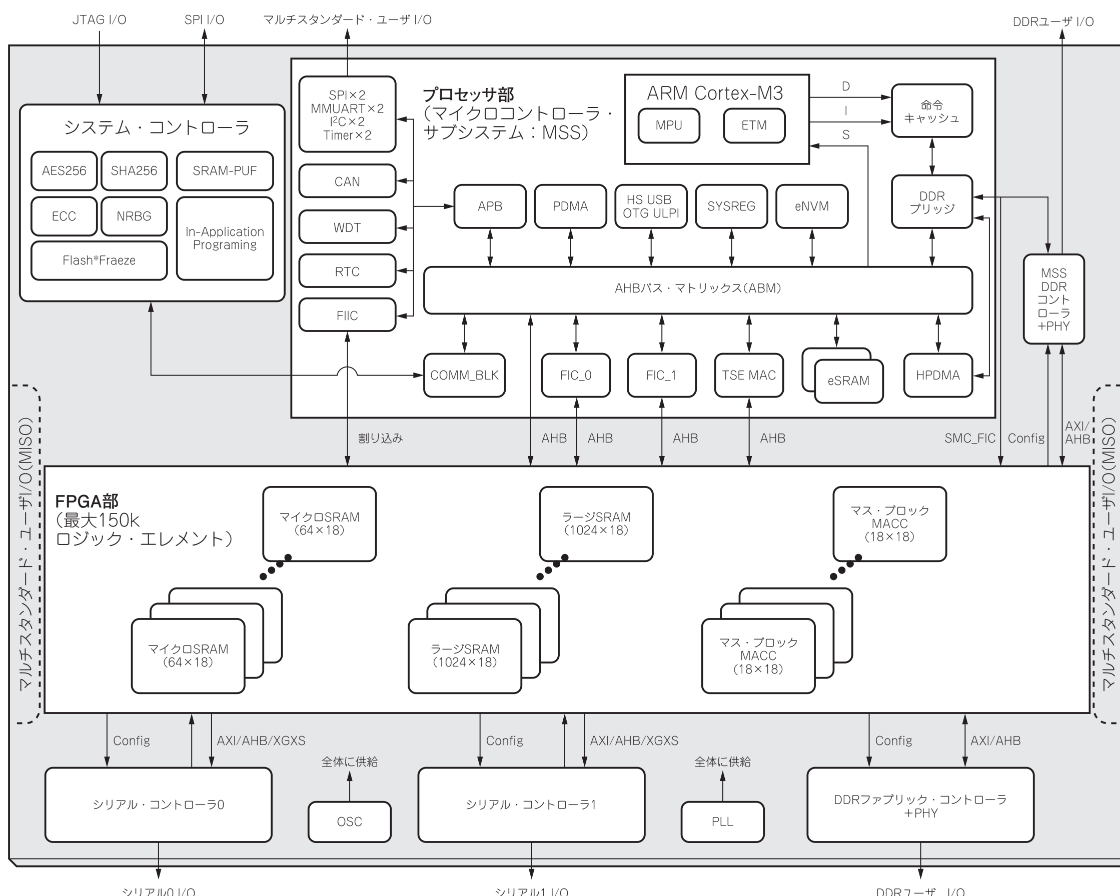
◆参考・引用*文献◆

(1) Getting Started Guide for the SmartFusion2 KickStart Kit version 1.0, September 2015, Avnet.
(2) SmartFusion2 KickStart Kit Schematic Rev. B, Jun 2015, Avnet.
(3) SmartFusion2 System-on-Chip FPGAs Product Brief Revision 23, April 2016, Microsemi.
(4) SmartFusion2 Microprocessor Subsystem User Guide Revision 11, February 2016, Microsemi.
(5) 浅井 剛；SmartFusion2で作るMyワンチップ・マイコン 次世代SmartFusion "SmartFusion2" の概要，FPGAマガジン，No.3，CQ出版社.

あさい・たけし

コラム ARM Cortex-M3マイコン・コア内蔵FPGA「SmartFusion2」

浅井 剛



図A [3] SmartFusion2の構成

● FPGA機能を持つCortex-M3マイコン

SmartFusion2の構成を**図A**に示します．FPGA部を除けば，各社から発売されているCortex-M3マイコンと大きな違いはありません．

FPGA部は，オンチップ・バスのAHB(Advanced High-Performance Bus)とファブリック・インターフェース・コントローラ(FIC)を介して接続されています．FPGA部のユーザ論理をAHBのマスタやスレーブとして使用できます．システムの処理性能を落とさずにユーザ論理を組み込めるというのは，プロセッサ内蔵FPGAならではの特徴です．

SmartFusion2は，FPGA部の論理規模によって7品種あります(**表A**)．論理規模に応じてメモリ容量等にも違いがあります．SmartuFusion2 KickStart Kitには，M2S010TQG144が搭載されています．

● 軍用でも使える高いセキュリティ

外部にコンフィグレーション・メモリを必要としないフラッシュ・アーキテクチャを採ります．FPGA部の回路情報やプロセッサのプログラム・コードに対するセキュリティ機能が組み込まれています．設計情報漏えいを気にするユーザにとっては大変魅力的だと思います(**表B**)．

Microsemi社は航空・宇宙分野を得意としており，一般的な商用(Cグレード)，産業用(Iグレード)に加え軍用(Mグレード)があります．

表A[3] **SmartFusion2のラインアップ**
論理規模は，DSPやメモリの使用によって変化する

<table>
<tr><th colspan="2">機　能</th><th>M2S005</th><th>M2S010</th><th>M2S025</th><th>M2S050</th><th>M2S060</th><th>M2S090</th><th>M2S150</th></tr>
<tr><td rowspan="3">ロジック/DSP</td><td>最大ロジック・エレメント（4LUT+DFF）</td><td>6,060</td><td>12,084</td><td>27,696</td><td>56,340</td><td>56,520</td><td>86,316</td><td>146,124</td></tr>
<tr><td>Mathブロック（18×18）</td><td>11</td><td>22</td><td>34</td><td>72</td><td>72</td><td>84</td><td>240</td></tr>
<tr><td>PLL & CCC</td><td colspan="2">2</td><td colspan="4">6</td><td>8</td></tr>
<tr><td rowspan="2">セキュリティ</td><td>AES256，SHA256，RNG</td><td colspan="7">各1</td></tr>
<tr><td>ECC，PUF</td><td colspan="4">—</td><td colspan="3">各1</td></tr>
<tr><td rowspan="6">MSS</td><td>Cortex-M3+命令キャッシュ</td><td colspan="7">あり</td></tr>
<tr><td>eNVM［Kバイト］</td><td>128</td><td colspan="4">256</td><td colspan="2">512</td></tr>
<tr><td>eSRAM［Kバイト］</td><td colspan="7">64</td></tr>
<tr><td>eSRAM［Kバイト］（Non-SECDED）</td><td colspan="7">80</td></tr>
<tr><td>CAN，10/100/1000イーサネット，HS USB</td><td colspan="7">各1</td></tr>
<tr><td>マルチモードUART，SPI，I²C，タイマ</td><td colspan="7">各2</td></tr>
<tr><td rowspan="3">ファブリック・メモリ</td><td>ラージSRAM（18Kブロック）数</td><td>10</td><td>21</td><td>31</td><td colspan="2">69</td><td>109</td><td>236</td></tr>
<tr><td>マイクロSRAM（1Kブロック）数</td><td>11</td><td>22</td><td>34</td><td colspan="2">72</td><td>112</td><td>240</td></tr>
<tr><td>全RAM［Kビット］</td><td>191</td><td>400</td><td>592</td><td colspan="2">1,314</td><td>2,074</td><td>4,488</td></tr>
<tr><td rowspan="3">ハイスピード</td><td>DDRコントローラ［チャネル数×幅］</td><td colspan="3">1×18</td><td>2×36</td><td colspan="2">1×18</td><td>2×36</td></tr>
<tr><td>SERDESレーン</td><td>0</td><td colspan="2">4</td><td>8</td><td colspan="2">4</td><td>16</td></tr>
<tr><td>PCI Expressエンドポイント</td><td>0</td><td colspan="2">1</td><td colspan="3">2</td><td>4</td></tr>
<tr><td rowspan="4">ユーザI/O</td><td>MSIO（3.3V）</td><td>115</td><td>123</td><td>157</td><td>139</td><td>271</td><td>306</td><td>292</td></tr>
<tr><td>MSIOD（2.5V）</td><td>28</td><td>40</td><td>40</td><td>62</td><td colspan="2">40</td><td>106</td></tr>
<tr><td>DDRIO（2.5V）</td><td>66</td><td>70</td><td>70</td><td>176</td><td>76</td><td>66</td><td>176</td></tr>
<tr><td>全I/O</td><td>209</td><td>233</td><td>267</td><td>377</td><td>387</td><td>412</td><td>574</td></tr>
</table>

表B　高いセキュリティ機能を持つ軍用グレードがある

オーダ・コード	サポートするセキュリティ機能		
	トランシーバ	データ	デザイン
ブランク	—	—	○
S	—	○	○
T	○	—	○
TS	○	○	○

表C　第2世代では高速化しているがアナログ機能がなくなった

仕　様	SmartFusion	SmartFusion2
製造プロセス	0.13 μm	65nm
Cortex-M3動作周波数	100MHz	166MHz
DDRメモリ・サポート	×	○
イーサネットMAC	10/100M	10/100/1000M
CANインターフェース	×	○
USBインターフェース	×	○
PCI Express エンドポイント・サポート	×	Gen2（×1/×2/×4，最大4個）
プログラマブル・アナログ・サポート	○	×

● 方向性…アナログ回路は削除して高速ディジタルFPGAに

SmartFusion2は，シリーズの中では第2世代の製品です．第1世代の製品との比較を**表C**に示します．

SmartFusion2は，製造プロセスが0.13 μmから一気に2世代（0.13 μm→90nm→65nm）微細化されたことにより，CPUはもちろん周辺機能も高速化され，新たにさまざまな高速シリアル・インターフェースを搭載しました．この反面，微細化が難しいアナログ回路が削除されています．高分解能のニーズが高まりますます搭載が厳しくなってきているアナログ回路は外部部品に任せ，その代わりさまざまな高速シリアル・インターフェースを内蔵するという方向へ方針を変更したということでしょう．

また基板レイアウトから考えてみれば，多ピンのアナログ機能搭載FPGAよりも，ディジタルFPGA＋外部のアナログ・ディジタルLSIの方が理想的な配置に近づけられますので，高速化も含めて考えるとMicrosemi社の方向性は理解できます．

ダウンロード・データあります

一人複数カメラの時代！ iPhoneで試して合点！

高性能カメラ探偵団

第5回 評価項目：解像度…レンズ性能を見極める定番指標

チャート：解像度

エンヤ ヒロカズ

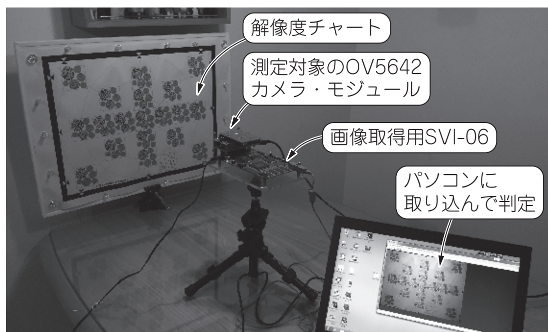


写真1　レンズの解像度をテストしている

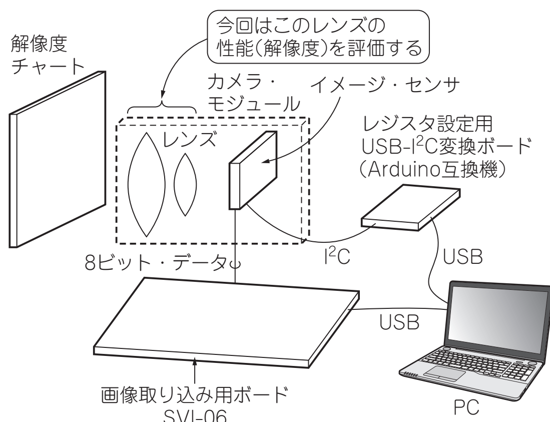


図1　レンズ解像度のテスト時の基板構成

今回からはレンズ性能を見極める指標の測定を行っていきます．今回はレンズの指標測定の定番となっている「解像度」です．テストの様子を**写真1**に，構成を**図1**に示します．解像度の測定には，一般的にはISO 12233規格準拠の解像度チャートが使われます．もちろんこのISOチャートを使うことは可能ですが，ちょっと使いにくい部分もあります．そこで今回は特製の解像度チャートを作成しました

解像度を表す指標

そもそも解像度とはどのような指標になっているのでしょうか．カメラで撮影したときにどこまで細かい被写体を再現できるかということなのですが，古くはアナログ・テレビ時代からTV本という単位で言及されることが多いです．レンズ単体では全て光学的に議論ができるので，MTF（Modulation Transfer Function）という単位で語られることが多いです．

しかし，カメラ・システムで考えると，解像度はレンズだけでは決まりません．イメージ・センサの画素数も大きな要素ですし，信号処理部分でのシャープネス処理などでも解像度は大きく変化します．そこでカメラ・トータルとしてはMTFではなくTV本が使われています．

● 指標1：TV本…カメラやテレビの解像度を表す

▶懐かしいアナログ・テレビのころに定義された

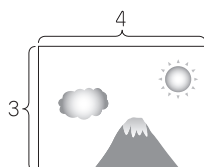
TV本はその名前の通り，テレビにおける走査線の本数がどこまで解像するかを示したものです．アナログ・テレビ時代はNTSCの走査線は525本でした．しかしインターレース伝送でしたので半分の262.5本をフィールド単位で伝送し，2フィールドで1フレームの画像を表現していました．そのため最大の本数は525本ですが，実力としてはそれよりも低くなってしまいました．

▶水平方向の解像度も縦方向と揃える

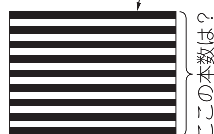
これは縦方向の解像度ですが，水平方向の解像度も同じように定義します．アナログ・テレビは水平/垂直のサイズが異なりアスペクト比は4：3です．そのまま画面全面での本数を決めてしまうと，水平と垂直で尺度が変わってしまいますので，水平方向も垂直と同じアスペクト（1：1）にします（**図2**）．

▶2本で1ペアそのペア数をTV本とする

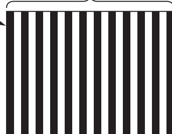
本数の数え方としては，2本でひとまとめ（ライン・ペア）という考え方をします（**図3**）．これは画像として1本分の走査線では解像しているか分からないためです．サンプリング定理として知られているシャノンの定理と同じ考えで，ある特定の空間周波数を再現す



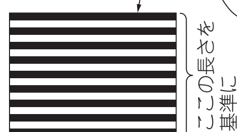
（a）通常の画像は縦横比が違う



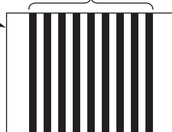
（b）垂直方向の走査線数を数えるとき



（c）水平方向の走査線数を数えるとき



（d）垂直方向の走査線を基準として…



（e）水平方向の走査線数を数える

図2　縦横比が異なるテレビ・モニタでは垂直方向の走査線数を基準に水平方向の走査線数を数える

るためには，その倍の空間周波数（画素数）が必要というわけです．資料のTV本はサンプリング周波数に相当する解像度になります．ペアでないと再現できないのにもかかわらず，指標はペアの一つ分というのはちょっと変な感じもしますが，このように定義されています．

● 指標2：MTF…レンズの解像度を表す

MTFはレンズ単体の解像度を語る場合に使います．イメージ・センサの画素数や信号処理は関係ありません．純粋に光学的にどこまで解像するかの関数として定義されています．関数なので数式で表現されるべきものではありますが，一般的には特性グラフで議論されることが多いです．例えば，横軸は画面内の位置（像高）で，縦軸はコントラスト比といった感じです．横軸は空間周波数の場合もありますが，縦軸はコントラスト比になります．

コントラスト比は白と黒のしま模様をレンズで投影したときに，出力の白黒の比が入力に対してどの程度落ちているかで表します．入力＝出力の場合はコントラスト比は1（理想値）になります．

MTFの特性が悪くなるとコントラスト比が落ちていき，完全にボケている状態は0になります．ではこ

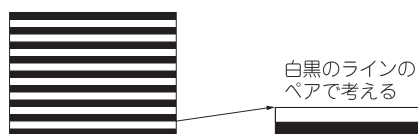


図3　TV本はライン・ペアの数で表現する

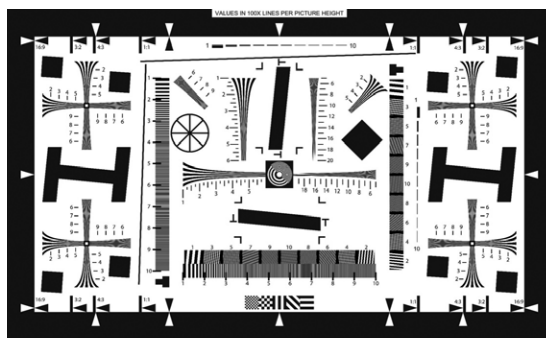


図4　有名なISO 12233規格の解像度チャート…初心者には扱いづらい

の0から1までの間で，果たしていくつの値ならば解像していると判断できるといえるでしょうか．人間の目は明暗の差は非常に敏感に感じますので，コントラスト比が0.2程度あれば解像していると判断してしまいます．もちろん個人差もありますので，正解はなく使用者が用途に応じてその都度判断しているという状況です．

初心者にも使いやすく！オリジナルCQ解像度チャート

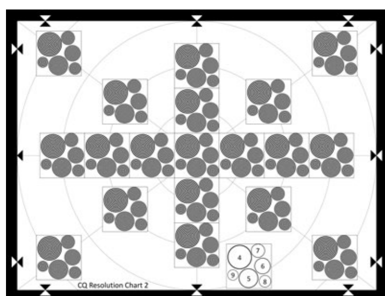
● 定番チャートの弱点

ダウンロード・データとして提供するCQ Resolution Chartについて解説します．解像度チャートしてはISO 12233規格の解像度チャートが有名です（**図4**）．しかし，この解像度チャートは使いにくいところがあります．まずはチャートの解像度が連続的に変化している点です．これは良い面もあり，100本以下の分解能で解像度を知ることができます．しかしどこまで解像しているかを判断するのは難しく，初心者には扱いづらいです．

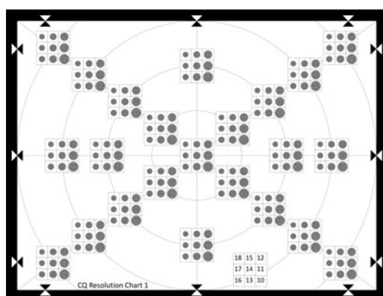
● 低解像度用と高解像度の両方を用意するチャートの特徴

▶その1：画素数に応じて2枚から選べる

今回のチャート（**図5**）は，空間周波数を一定のパターンにすることによって解像度の判定がしやすくなっています．解像度は400～1,800TV本まで100TV本単位で配置しています．またスペースの都合で400～900TV本と1000～1,800TV本の2枚に分割しています．解像度はカメラ・モジュールの場合はまず



（a）低解像度400～900本用



（b）高解像度1,000～1,800本用

図5　初心者にも使いやすく！実験に用いたオリジナル解像度チャート

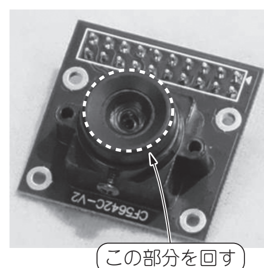


写真2　OV5642カメラ・モジュールのフォーカスはレンズを回すことで調整できる

イメージ・センサの解像度が支配的になりますので，画素数に応じて最適なチャートを選ぶと良いでしょう．

▶その2：中心と周辺の解像度ムラをチェックできる

次に画面内に可能な限り同じパターンを配置しています．これは画面中心と周辺で解像度が異なる場合が多いためです．ISO 12233チャートは画面内に配置されているチャートが少ないので，所望の位置での解像度測定ができません．また画面内位置の目安として画像中心から20％単位で同心円を描いています．画面内の中心からの距離を知りたいときにも便利です．次に今回のチャートの一番大きな特徴は形状を円形にしたことです．

ISO 12233チャートでは，水平，垂直，45°方向のパターンが用意されていますが，画面内での位置が異なります．またレンズと特性としては，同心円方向と像高方向の二つの方向で語るのが光学的には合っています．

このような問題を解決するのが円形状のチャートになります．円形チャートであれば，被測定位置の近傍で水平，垂直，同心円，像高のすべての方向の解像度が測定可能です．

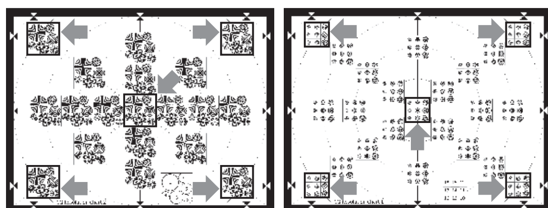
評価実験

● ステップ1：チャートの撮影

解像度を測定するためには，フォーカスが合った状態で撮影する必要があります．本連載で用いているOV5642カメラ・モジュールはレンズ部分を回転させることにより，フォーカス調整が可能です（**写真2**）．また，比較として用いているiPhone 6は10cm程度まで近づいて撮影できます．そこで今回はA2サイズに印刷したチャートを用いて評価を行いました．被写体距離はOV5642で40cm，iPhone 6で42cmとなりました．もちろんフォーカスを合わせた状態です．

今回は解像度の測定なので，十分に明るい環境下であれば，光源の照度はあまり厳密に管理する必要はありません．筆者も自宅の居間で通常の照明を用いて撮影しました（**写真1**）．露光もホワイト・バランスもオート設定です．

OV5642は5M JPEG出力の取り込みができないので，安定して出力できるSXGAでの解像度での撮影となりました．iPhone 6はデフォルトの8Mピクセルで撮影しました．測定位置は，画像中央と周辺（像高7割付近）の解像度を測定します．周辺は上左，上右，下左，下右の4カ所，合計5カ所の値を測定します（**図6**）．



（a）400～900本用　（b）1,000～1,800本用

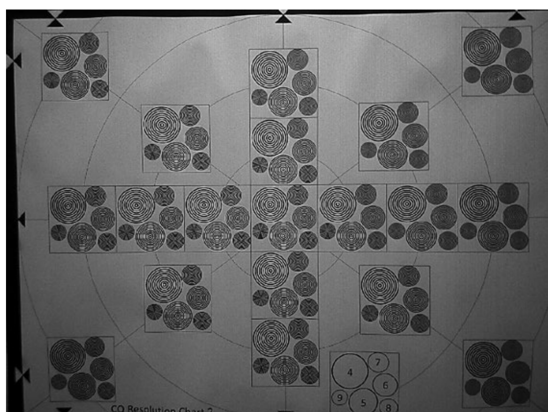
図6　測定に使ったチャート内の場所

● ステップ2：解像の判断

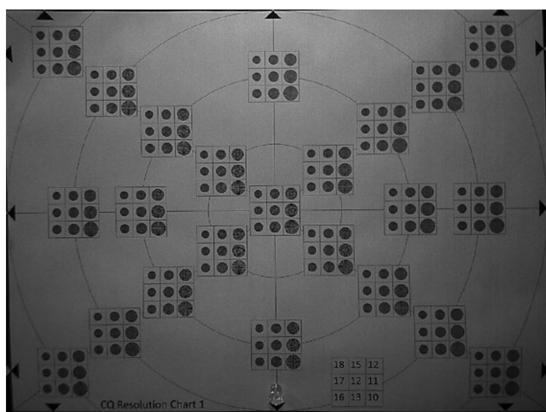
撮影結果を**図7**に示します．データ処理ですが，画像ビューワや画像編集ソフトウェアで画面内の任意の位置の解像度パターンを拡大して，解像しているかを判断します．ただし，前述の通り解像しているかの判断は非常に難しいです．今回は筆者の判断を**表1**に示します．もちろん個人差がありますので，あくまで参考にとどめていただければと思います．

▶OV5642カメラ・モジュール

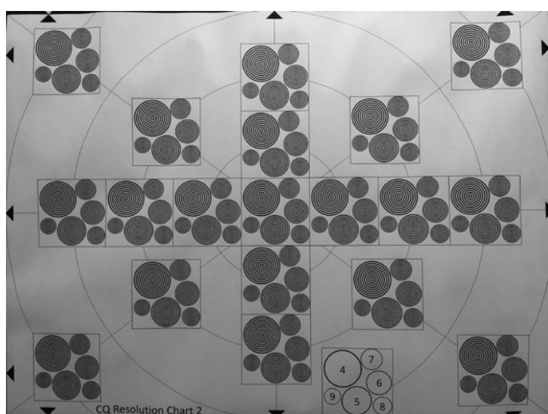
OV5642の画像を見ると，高解像度のチャート［**図7(b)**］は全く解像していないことが分かります．また低解像度のチャート［**図7(a)**］の方も，信号処理の問題なのか，折り返しノイズのようなモアレ状のノイズが発生しており，解像しているか判定が困難でした．辛うじて700本程度でしょうか．また*SN*測定のとき



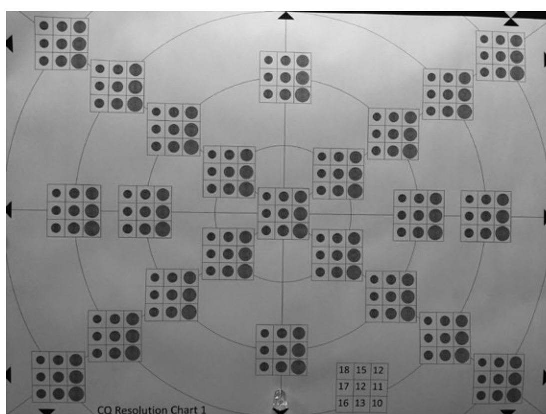
(a) OV5642 400～900本用



(b) OV5642 1,000～1,800本用



(c) iPhone6 400～900本用



(d) iPhone6 1,000～1,800本用

図7　チャートを撮影して得られた画像

にも問題になりました「縦筋状のノイズ」も出ています．ノイズ成分は解像度の測定結果には大きな影響は与えませんが，信号処理的に問題があるのかもしれません．結果として折り返しノイズになってしまっていると思われます．また左の方が右に対して解像が良いです．原因はレンズの傾きや，ねじ込み部分のガタつきなどが考えられます．また水平/垂直の測定結果はほぼ同じでした．

▶iPhone 6

iPhone 6はもともと8Mピクセルで撮影していることもあり，中心は1,800本程度，周辺も1,500本程度と非常に良い値を示しています．また周辺のばらつきも少なく，iPhoneはレンズの性能も一味違うということが分かります．

● 考察

OV5642は解像度が低い方が支配的になってしまい，レンズの性能を見られる状況になっていません．今後5M JPEGの撮影ができるようになったら，再度測定してみる予定です．

表1　解像度の判定結果（H：水平方向，V：垂直方向）

カメラ	H/V	中　央	右　上	左　上	右　下	左　下
OV5642	H［本］	700	500	600	500	600
	V［本］	700	500	600	500	600
iPhone 6	H［本］	1,800	1,500	1,500	1,500	1,500
	V［本］	1,800	1,500	1,500	1,500	1,400

iPhone 6の方はカタログの画素数に相当する数字でした．周辺では若干落ちますが，四隅で均等な落ち具合であり，実用上は問題ないかと思います．

ダウンロード・データの入手先

http://www.cqpub.co.jp/interface/download/contents.htm

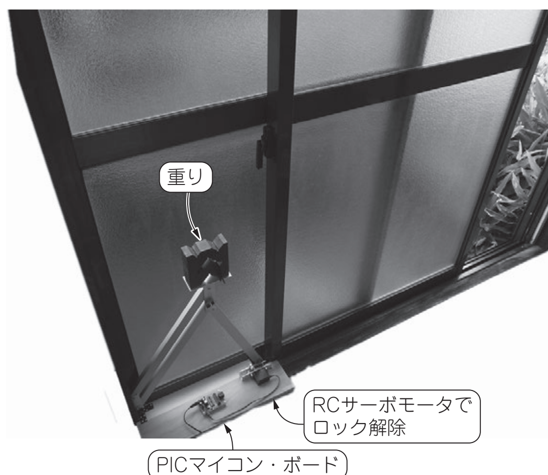
えんや・ひろかず

夏のビギナ企画！

パルスを送れば向き自由自在！

動きものビギナのための RCサーボモータ入門

川村 聡



（a）開



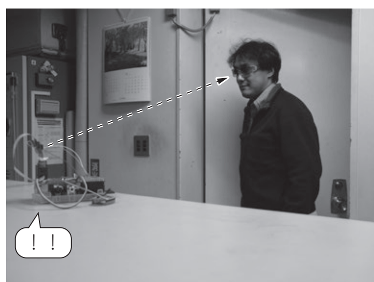
（b）閉

写真1[1]　**動きものが作れたら広がる世界1…窓閉め装置**
アームの保持（ロック）に

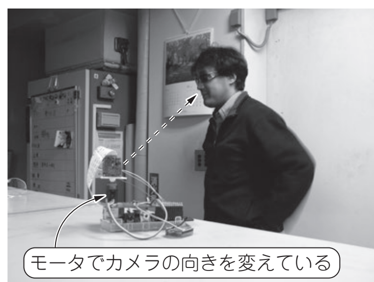
動きのある装置を作る際に，簡単に利用できるのがRCサーボモータです．例えば，**写真1**[1]や**写真2**[2]のような装置がモータの専門家でなくても作れます．IoT時代はリアルな物を動かせることがますます重要になると思います．そこで本稿ではモータ・ビギナ向けにRCサーボモータの基本的な使い方を紹介します．　**（編集部）**

楽チン！RCサーボモータ

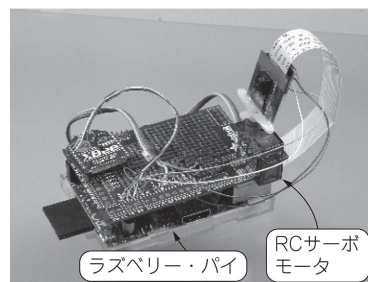
RC（ラジコン）サーボモータは，**写真3**のような外観で，操縦プロポ（RC模型を操縦するための操作量を電波に変えて送る）のスティック角に連動して動くアクチュエータ・モジュールです．



（a）カメラが侵入者を検出



（b）自動追尾中



（c）顔追跡カメラのハードウェア

写真2[2]　**動きものが作れたら広がる世界2…自動追尾カメラ**
不審者を追いかける



写真3　RCサーボモータの例
Micro-2BBMG（Grand Wing Servo-Tech）

● モータから減速ギアまで全部入り

内部ブロック図を**図1**(**a**)に示します．RCサーボモータ内にはDCブラシ付きモータ，減速ギヤ，制御基板，位置検出器などが内蔵されおり，サーボホーンと呼ばれる出力軸を±90°の範囲で，任意の角度に止めることができます．

● マイコンから目標位置パルスを送るだけ

図1(**b**)に示す受信機を使わず，マイコンから直接ドライバICを制御できるため，例えば多数のRCサーボモータをマイコンで制御して，ロボットの関節のような動きをさせることも可能です．マイコン基板として筆者は「RX220マイコンボード開発セット」（秋月電子通商，20MHz動作）を使っています．

● 必要な線は3本だけ

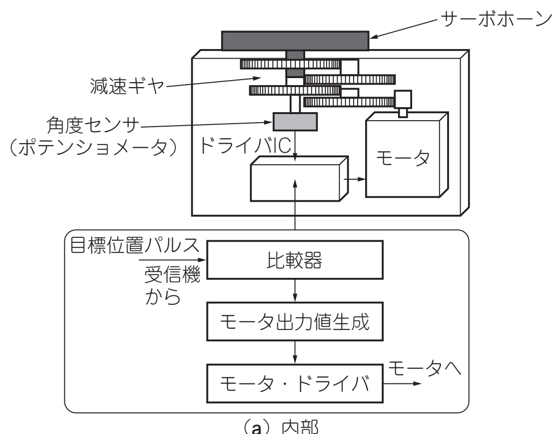
RCサーボモータから出ているハーネスは電源，GND，コントロール信号の3本だけで，ピン配置は各社ほぼ同じになっているようです．どれが何のピンかは，ハーネスの色で見分けられます（**図2**）．ただし，古い製品や海外品などでは，ピン配置が異なります．

選び方

RCサーボモータの仕様としては，トルク，スピード，サイズ，重量などがあります．

▶トルク［kg・cm］

最大トルク＝モータ停動時（指定した角度を保つ）のトルクを表しますが，このトルクで使い続けると壊れる可能性があります．



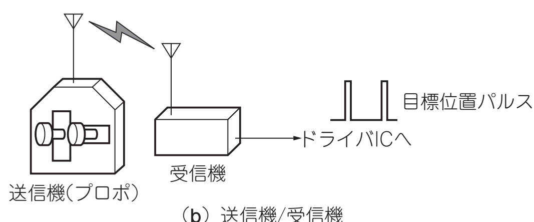


図1　RCサーボモータはDCブラシ付きモータとモータ・ドライバ/角度センサ/減速ギヤで構成される

▶スピード［sec/60deg］

無負荷時，ホーンが60°回るのにかかる時間です．

▶サイズや重量

基本的にはサイズが大きいものはトルクが大きいです．ギヤが金属製のものは重いですが，そのぶん耐久性が高く壊れにくい特徴があります．

▶ギヤの材料

RCサーボモータは基本的には玩具/ホビー用ですが，用途ごとにもう少し細かく分けると，飛行機用，カー用，ロボット用があります．飛行機用は内部のギヤがプラスチックでできていて軽量です．カー用は金属製のギヤやベアリングを使用した堅牢なものが多い

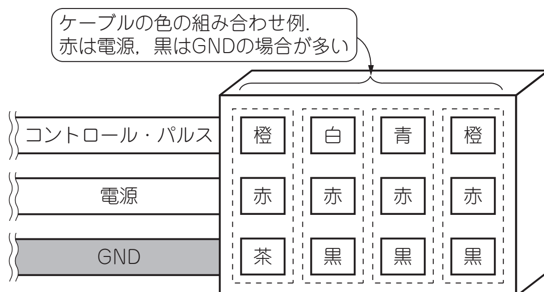


図2　接続のコネクタのピン配置はどの製品も共通
ただし海外製は除く

です．ロボット用は高トルク/高速/高精度ですが，価格も大幅に高くなります．

動かすために

● ハードウェア…電源はマイコン用と共通でOKだが突入電流による電圧降下の対策が必要

RCサーボモータとマイコンとの相性は抜群です．外付け回路や通信処理が不要で，出力ポート1本を接続するだけです．**図3**にRCサーボモータの駆動回路例を示します．

ほとんどのRCサーボモータは4.8～6V程度で動作するため，マイコンが5V動作する場合は電源を共通にできます．ただし，RCサーボモータの起動時の消費電流が大きい場合，マイコンの電源電圧が瞬間的に低下し，マイコンにリセットがかかったり動作が不安定になったりする可能性があります．この対策として，動作電圧範囲の広いマイコンを使用することと，**図4**のように，マイコン側の電源回路にパスコンとダイオードを追加することなどが有効です．

● ソフトウェアの決まりごと

RCサーボモータを動かすソフトウェアとしては，角度指令値を簡単なパルス指令として一方的に送るだけで，面倒な通信処理やアナログ入出力処理などは必要ありません．マイコン導入時のLED点滅テスト（通称Lチカ）と同等レベルの処理で動かせます．

RCサーボモータを所定の位置まで回すには，**図5**に示すように，“H”パルスの幅が変動する一定周期のPWM（Pulse Width Modulation）信号を入力します．サーボホーンが止まる角度は入力したパルスの“H”時間の幅（多くは0.6～2.2msの範囲）に比例します（**図6**）．

“H”と“L”時間を合わせたサーボの制御周期は，通常は10～20msの範囲で一定に取りますが，この周期ごとに角度指令値が更新されるため，滑らかな連続動作，機敏な反応を得るためにはなるべく短くしたいところです．ただし，安価なサーボではあまり周期を短くしすぎると制御回路が応答しなくなります（高速のパルス入力に対応した特別なサーボもある）．

● 駆動プログラム1…タイマとポーリングでI/OピンをON-OFF

RCサーボモータ駆動信号はタイマ機能を利用して作るのが最も簡単です．**リスト1**にプログラム例を示します．Lチカ同様，タイマの経過時間待ちでパルスエッジを上下するだけのシンプルなものです．

サーボの動作角はパルス幅に比例するため，比例定数やオフセット量を定義して，サーボの型番ごとに合わせ込んでおけば，角度をそのまま指令値に設定できます．**リスト1**のプログラムでは，16ビットのタイマ（CMT3）を使い，0.1°単位でサーボ指令値を生成でき

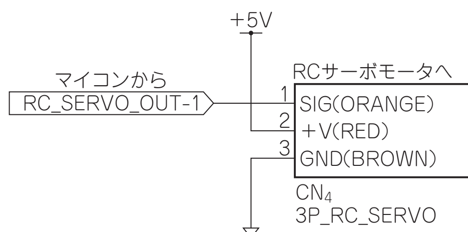


図3　RCサーボモータの駆動回路

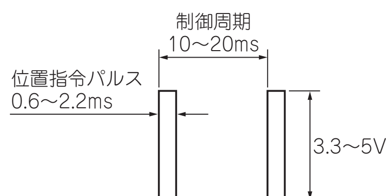


図5　位置指令のパルス時間は決まっている

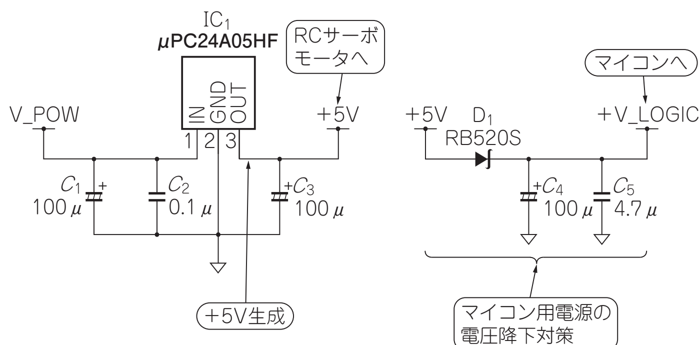


図4　マイコンの電源には一工夫必要

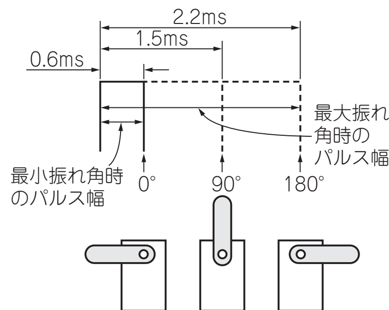


図6　欲しい角度とパルス時間との関係

るようにしていますが，たいていのRCサーボモータは±1°程度の誤差を持っているため，残念ながら0.1°の精度は期待できません．

● 駆動プログラム2…タイマ割り込みでI/OピンをON-OFF

他のマイコンやPCと通信しながらRCサーボモータを駆動したい場合や，センサ入力に応じてその都度サーボの指令角度を変更したい場合など，RCサーボモータ駆動信号の生成と並列に別のプログラムを走らせたいときには，**リスト1**のようにソフトウェアのポーリングでは効率が悪いので，タイマ割り込みを使います．

リスト2が**リスト1**をタイマ割り込み仕様に変更したプログラムです．割り込み関数内での処理は，エッジの上げ下げと次の割り込みタイミングの設定だけなので，非常に軽い処理で済みます．また，割り込みを使えばRCサーボモータ指令角の更新タイミングを本ループの処理と分離できるため，例えばPCとの通信処理と組み合わせた場合，通信は20ms周期で行い，サーボ指令値更新は10ms周期で行うといったことが容易になります．

● 駆動プログラム3…PWM出力ピンを使う

RCサーボモータの駆動パルス生成に，モータ制御用のマイコンに内蔵されているPWM出力機能を使えば，同じ角度を保持したい場合などにソフトウェア・リソースを一切消費せずに済みます．

リスト3は，RX220に内蔵されている8ビット・タイマ(TMR3)を使ったプログラム例です．二つのコンペア・マッチ・レジスタに値を代入しておけば，PWM出力用のピンからサーボ駆動信号が連続的に送出されます．

ただし，**リスト3**のプログラムには一つ問題があって，サーボの角度分解能が約28°刻みとかなり粗くなります．これは，角度指令パルスの変動幅がPWM-Duty換算で4～15%程度と，サーボ周期全体に対して短いためです．本来は可変したいパルス幅の範囲(0.6～2.2ms)に，タイマ・カウンタのレンジを設定したいところですが，通常のマイコン内蔵のPWM機

リスト1　駆動プログラム1　タイマとポーリングでI/OピンをON-OFF

CPUクロックは内蔵高速発振回路32MHzで，PCLK=32MHzとする．CPUやシステム初期化関数はここには記述していないため，秋月電子通商のサンプル・プログラムやルネサス社RX220のハードウェア・マニュアルなどを参照のこと

```
//************グローバル変数***********************
int RC_SERVO_RANGE=620;
int RC_SERVO_PERIOD=92;
int RC_SERVO_OFFSET=15;
//サーボ用タイマ(CMT3)を設定しカウント・スタート
/*
  PCLK=32MHzを1/32分周 1カウント=1usec
  16bitタイマのためオーバフローは65535usec=約65msec
*/
void start_servo_cycle(void)
{
  SYSTEM.MSTPCRA.BIT.MSTPA14=0;
  //モジュールストップコントロールを解除(CMT2,CMT3)
  CMT.CMSTR1.BIT.STR3=0;  //カウントを停止
  CMT3.CMCR.BIT.CKS = 1;
  //タイマクロック選択(0**1/8 1**1/32 2**1/128 3**1/512)
  CMT3.CMCR.BIT.CMIE = 0;  //割り込みを行うかどうか
  CMT3.CMCNT = 0;         //タイマカウンタの初期化
  CMT3.CMCOR = 0xFFFF;
  //コンペアマッチクリアのタイマカウンタ
  CMT.CMSTR1.BIT.STR3=1;    //カウントを開始
}
//*サーボ信号のH期間が経過するのを待つ(usecカウンタ)
void wait_servo_pulse(int usec)
{
  while(CMT3.CMCNT<usec){}
  //タイマカウンタ値が設定値になるのを待つ
}
//サーボタイマ制御周期の経過待ち(msecカウンタ)
void wait_servo_cycle(int msec)
{
  while(CMT3.CMCNT<msec*1000){}
  //タイマカウンタ値が設定値になるのを待つ
}
//サーボを1個駆動する
/*
  引数の角度指令値[angle_01]は0.1deg単位で指定 982→98.2[deg]
  サーボ振れ角は以下のグローバル変数で調節
  (RCサーボの型番：GWS Micro-2BBMG の場合)
  RC_SERVO_OFFSET  620    オフセット
  RC_SERVO_RANGE  92    レンジ
  RC_SERVO_PERIOD  15        制御周期[ms]
*/
void Drive_Servo_Simple(int angle_01)
{
  unsigned int h_span;
  h_span=RC_SERVO_OFFSET+angle_01*RC_SERVO_
                      RANGE/100;  //位置指令のHパルス幅を計算する
  start_servo_cycle();
  //サーボ制御周期をカウントアップ開始
  PORTH.PODR.BIT.B3=1;  //サーボ信号パルスをHへ
  wait_servo_pulse(h_span);
  //角度MINに当たる時間のウェイト
  PORTH.PODR.BIT.B3=0;  //サーボ信号パルスをLへ
  wait_servo_cycle(RC_SERVO_PERIOD);
  //サーボ制御周期が終了するのを待つ
}

void RC_Servo_Drive_Test_1(void)
{
  int i,total,angle;
  RC_SERVO_OFFSET=620;
  RC_SERVO_RANGE=92;
  RC_SERVO_PERIOD=15;

  total=100;  //繰り返し回数 15msx100回=1.5s
  while(1){
  angle=1350;  //135degを指定
  for(i=0;i<total;i++)
  Drive_Servo_Simple(angle);
  //サーボパルスを一定回数出力する

  angle=450;  //45degを指定
  for(i=0;i<total;i++)
  Drive_Servo_Simple(angle);
  //サーボパルスを一定回数出力する
  }
}
```

リスト2　駆動プログラム2　タイマ割り込みでI/OピンをON-OFF

```
//**********グローバル変数*************************
int RC_SERVO_RANGE=620;
int RC_SERVO_PERIOD=92;
int RC_SERVO_OFFSET=15;
int RC_SERVO_ANGLE=900;
int RC_SERVO_H_SPAN=0;
int RC_SERVO_L_SPAN=0;
/*割り込みでサーボパルスを生成*/
void Init_CMT3_for_RC_Servo(void)
{
  int ini_h_span;
  ini_h_span=RC_SERVO_OFFSET+RC_SERVO_ANGLE*
            RC_SERVO_RANGE/100;  //位置指令のHパルス幅を計算する
  RC_SERVO_H_SPAN=ini_h_span;
  //グローバル変数にコピーしておく
  SYSTEM.MSTPCRA.BIT.MSTPA14=0;
  //モジュールストップコントロールを解除(CMT2,CMT3)
  CMT.CMSTR1.BIT.STR3=0;  //カウントを停止
  CMT3.CMCR.BIT.CKS = 1;  //タイマクロック選択(0**1/8
                              1**1/32 2**1/128 3**1/512)
  CMT3.CMCR.BIT.CMIE = 1;
  //コンペアマッチ割り込みを行うかどうか
  CMT3.CMCNT = 0;  //タイマカウンタの初期化
  CMT3.CMCOR = ini_h_span;
  //コンペアマッチ用のカウンタ
  PORTH.PODR.BIT.B3=1;  //サーボ信号パルスをHへ

//***********割り込み設定**************************
  ICU.IER[3].BIT.IEN7=1;
  //CMT割り込みイネーブルビット
  ICU.IPR[7].BIT.IPR=7;
  //割り込み優先度レジスタ(0******最低   7******最高)
  CMT.CMSTR1.BIT.STR3=1;  //カウントを開始
}
/*割り込み関数処理
  ①サーボ信号のエッジ極性を反転する
  ②次の割り込み発生タイミングを設定(次がHの場合は角度指令値,
                        次がLの場合はサーボ周期-H期間)*/
void Excep_CMT3_CMI3(void)
{
  PORTH.PODR.BIT.B3^=1;  //パルスの極性反転
  RC_SERVO_H_SPAN=RC_SERVO_OFFSET+RC_SERVO_ANGLE*
                      RC_SERVO_RANGE/100;  //H期間のカウンタ値
  RC_SERVO_L_SPAN=RC_SERVO_PERIOD*1000-
                    RC_SERVO_H_SPAN;       //L期間のカウンタ値
  if(PORTH.PODR.BIT.B3==0)  //パルスがHの場合
  CMT3.CMCOR = RC_SERVO_L_SPAN;
  //次のコンペアマッチをL期間に設定
  else  //パルスがLのとき
  CMT3.CMCOR = RC_SERVO_H_SPAN;
  //次のコンペアマッチをH期間に設定
  ICU.IR[31].BIT.IR = 0;
  //割り込みステータスフラグをクリア
}
void RC_Servo_Drive_Test_2(void)
{
  RC_SERVO_OFFSET=620;
  RC_SERVO_RANGE=92;
  RC_SERVO_PERIOD=15;
  RC_SERVO_ANGLE=900;
  //初期の指令角度をグローバル変数に代入
  Init_CMT3_for_RC_Servo();
        //割り込みを起動(以後,割り込みを止めるまでパルス出力を続ける)
  while(1){
  RC_SERVO_ANGLE=450;
  //指令角度をグローバル変数に代入
  wait_msec(1500);  //時間待ち
  RC_SERVO_ANGLE=1350;
  //指令角度をグローバル変数に代入
  wait_msec(1500);  //時間待ち
  }  //無限ループ
}
```

リスト3　駆動プログラム3　PWM出力ピンを使う

```
//**********グローバル変数*************************
int RC_SERVO_RANGE=620;
int RC_SERVO_PERIOD=92;
int RC_SERVO_OFFSET=15;
int RC_SERVO_ANGLE=900;
int RC_SERVO_H_SPAN=0;
int RC_SERVO_L_SPAN=0;
/*
  TMR3をPWM出力に設定し,サーボ信号を生成
  サーボ周期を生成するのに十分な分周比を選択
                    (1/1024で8msec 1/8192で65msecまで測れる)
  CCLR=1にセットしてコンペアマッチAでカウンタクリア設定する
                                        (TCORAが周期レジスタ)
  コンペアマッチAの極性をH,コンペアマッチBの極性をLにセット
  コンペアマッチカウンタAにサーボ制御周期に相当する値を代入
  コンペアマッチカウンタBにサーボ指令角度に相当する値を代入
*/
void Init_TMR3(void)
{
  SYSTEM.MSTPCRA.BIT.MSTPA4=0;
  //モジュールストップを解除(TMR2,TMR3)
//**********ピンの設定*****************
  PORT5.PDR.BIT.B5=1;  //P55を出力ピンにセット
  MPC.PWPR.BIT.B0WI=0;  //PFSWEビットの書き込みを許可
  MPC.PWPR.BIT.PFSWE=1;  //PFSレジスタへの書き込みを許可
  MPC.P55PFS.BIT.PSEL=5;  //PC55をTMO3
  MPC.PWPR.BIT.PFSWE=0;
  //PFSレジスタへの書き込みを禁止
  PORT5.PMR.BIT.B5=1;  //P55を周辺機能にセット
//*******タイマ機能の設定******************
  TMR3.TCNT     =0;  //タイマカウンタをクリア
  TMR3.TCR.BIT.CCLR  =1;
 //カウンタクリアビット(0**禁止 1**コンペアマッチA 2**B 3**外部)

  TMR3.TCCR.BIT.CKS  =6;
  //クロック(分周比 1/8192 256usec刻み  最大65msec)
  TMR3.TCCR.BIT.CSS  =1;
  //クロックソース(内部クロック)
  TMR3.TCSR.BIT.OSA  =2;  //コンペアマッチAの極性=H
  TMR3.TCSR.BIT.OSB  =1;  //コンペアマッチBの極性=L
  TMR3.TCORA=59;  //サーボ周期をカウンタAにセット
  TMR3.TCORB= 6;  //サーボ指令角をカウンタBにセット
}

//角度指令値をタイマカウンタ値に変換してPWMを更新
void Set_Servo_Angle_PWM(int angle)
{
  int h_span;
  h_span=RC_SERVO_OFFSET+angle*RC_SERVO_RANGE/100;
                                          //H期間[usec]
  h_span/=256;  //タイマカウント値に変換
  TMR3.TCORB=h_span;
  //サーボ指令角をカウンタBにセット
}

//************PWMを使ったサーボ出力***************
void TEST_RC_SERVO_PWM(void)
{
  RC_SERVO_OFFSET=620;
  RC_SERVO_RANGE=92;
//  RC_SERVO_PERIOD=15;
//  RC_SERVO_ANGLE=900;
    //初期の指令角度をグローバル変数に代入

  Init_TMR3();
  while(1){
  Set_Servo_Angle_PWM(450);  //サーボ角度変更
  wait_msec(1500);  //時間待ち
  Set_Servo_Angle_PWM(1350);  //サーボ角度変更
  wait_msec(1500);  //時間待ち
  }
}
```

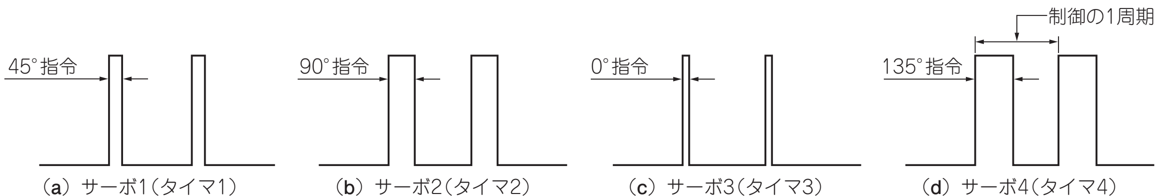


図7　RCサーボモータを4個同時に駆動する場合はタイマも4チャネル必要

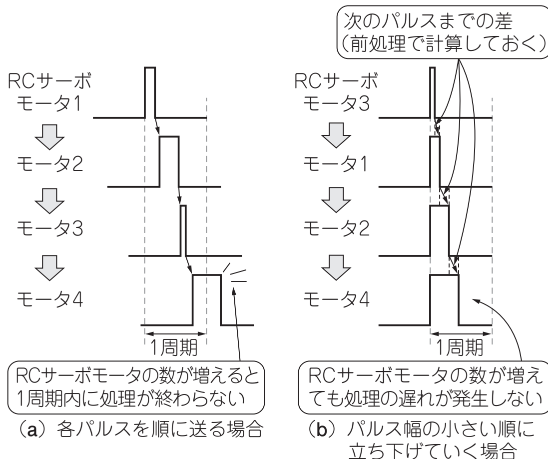


図8　タイマ・カウンタを1チャネルで済ませる方法

能は，デューティ比を0～100%で可変する前提になっているため，タイマ・カウンタも0～100%の範囲(＝サーボ周期)に設定せざるを得ません．その結果，例えば8ビットのPWMを使用した場合，サーボ角指令値の分解能はサーボ動作範囲(=±90°)の8ビット＝256段階ではなく，せいぜいその11%程度(=256×0.11≒28段階)になってしまいます．このため，もしPWM機能を利用してRCサーボモータの角度を細かく制御したい場合は，16ビットや32ビットのPWMを使う必要があります．

ロボットではあるある！ 複数のRCサーボモータを制御する

● タイマ・カウンタが足りない

複数のRCサーボモータを使った多軸制御を行いたい場合，**図7**のようにRCサーボモータの数だけ独立したタイマ・カウンタ(もしくはPWM出力)を使って駆動信号を生成します．しかし，例えば10個のRCサーボモータを同時に制御したい場合は，マイコンに内蔵するタイマのチャネル数が不足することから，一つのタイマで複数のRCサーボモータ駆動信号を生成するための工夫が必要になります．

図8(a)のように一つのタイマを使って順次指令パルスを送り出す方法では，RCサーボモータの数が多いと制御周期が長くなり，例えば10個の制御では，最大で2.2ms×10＝22msになります．また，指令値が更新されるタイミングがRCサーボモータごとにずれてしまうため，多軸の動作を正確に同期させられません．例えばRCサーボモータをロボットの関節に使う場合だと，関節1に対して，関節10の動作タイミングが最大22ms遅れてしまいます．

● 一つのタイマで複数のパルスを作る

これを解決するため**図8(b)**のようなパルスを生成します．まず駆動指令を送る前に，あらかじめ角度指令パルスを幅の小さい順にソートし，各パルス幅の差を計算しておきます(前処理)．そして全チャネルを同時に“H”にします．その後，短い指令パルスのものから順次“L”にしていきます．前処理の計算を本ループ内で行い，エッジの上げ下げはタイマ割り込みで行うようにします．この方法ではデータのソートなどで前処理に多少時間がかかりますが，この前処理の時間がRCサーボモータの最小パルス幅(約0.6ms)に収まる限り，いくら軸数が増えても一つのタイマで多数のRCサーボモータ信号を生成できます．

◆参考・引用*文献◆

(1) *中嶌 浩之，川村 聡；特集「全集付き！IoTサーバ使い放題！データ×ネット入門」，第3章　IoTの実験1…Linuxボード×リアルタイム気象データ，Interface，2016年1月号，CQ出版社．

(2) *森岡 澄夫；特集「ラズベリー・パイ×カメラで本格派画像処理」，第8章　Linuxでそこそこすばやく I/Oするテクニック，Interface，2014年1月号，CQ出版社．

かわむら・さとし

個人で試せる！

ダウンロード・データあります

キット発売中！

生体(バイタル)センシング実験室

第10回 静電容量方式生体センシングのキモ…ソフトウェア制御周波数発振器

上田 智章

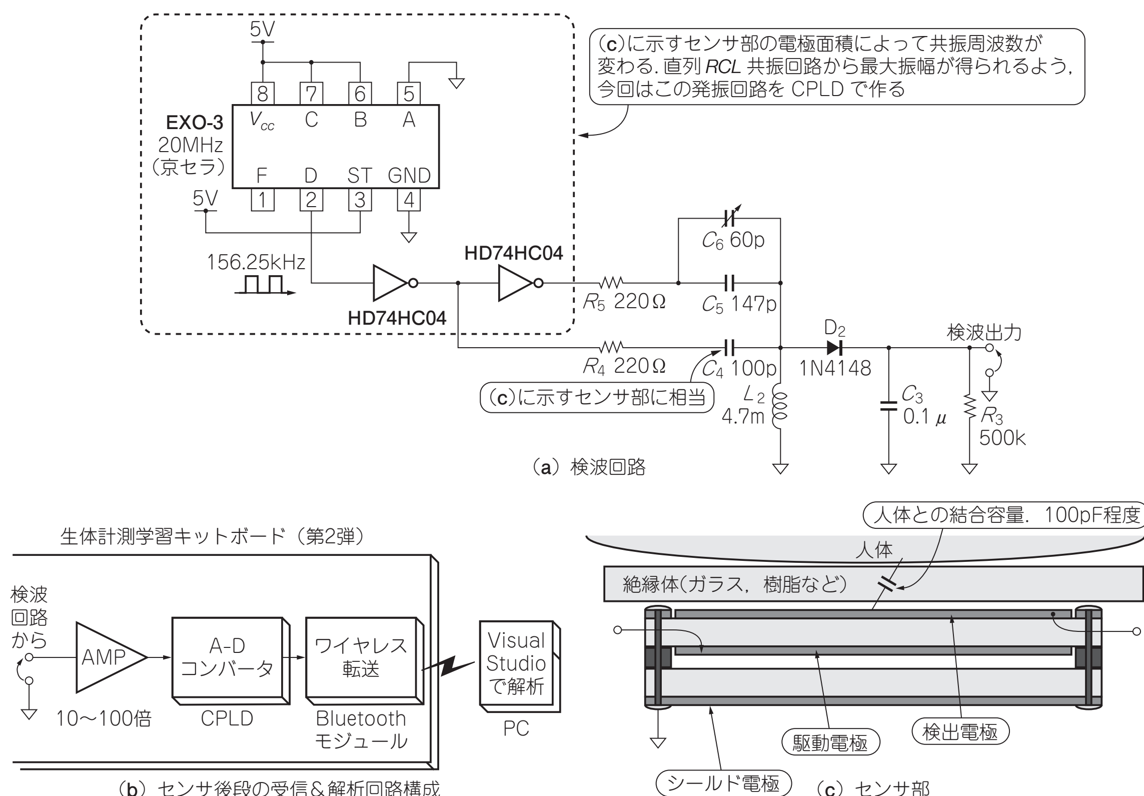


図1　人体との距離によって静電容量が変わることを利用し呼吸や心拍の変動を検出する静電容量型センサ

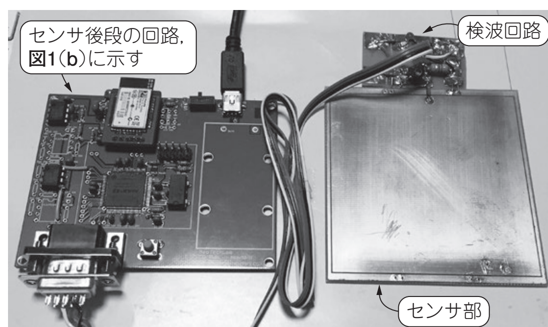


写真1　静電容量を検出するセンサ

本誌2016年7月号と9月号で紹介した静電容量型センサ（**図1**，**写真1**）は，椅子の背もたれやふとんの裏側に仕込んで使うことを想定しています．人体のどの部分の生体組織誘電率の変化を測定するかで，電極面積が変わります．そして，その電極面積に応じて共振周波数が大きく変化してしまうので，駆動周波数の方をソフトウェアにより最適な周波数に変化させる必要が生じます．そこで直列*RCL*共振回路を駆動する周波数を，「可変周波数発振器」によって最適値にキャリブレーションする必要があります．

今回はこの可変周波数発振器をCPLD（Complex Programmable Logic Device）で作ります．CPLDに

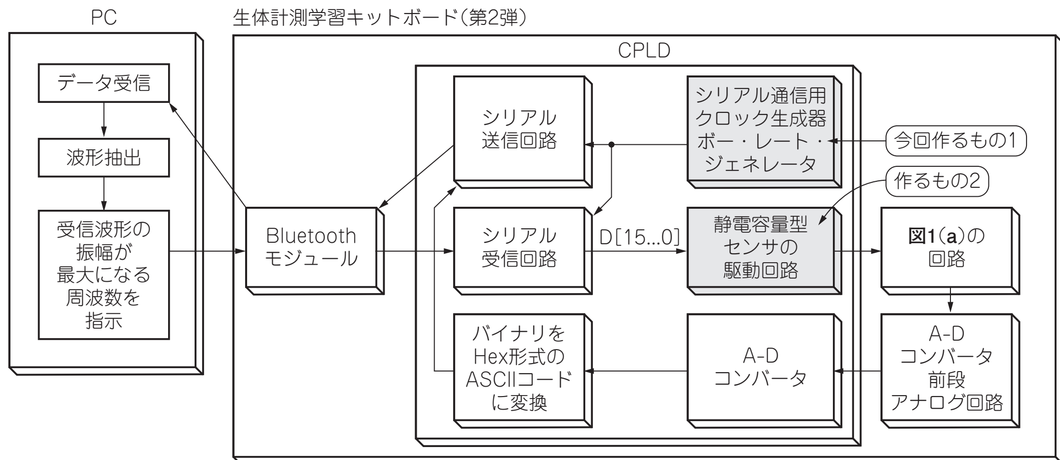


図2　今回作るもの…静電容量型センシングのキモとなる可変周波数発振回路

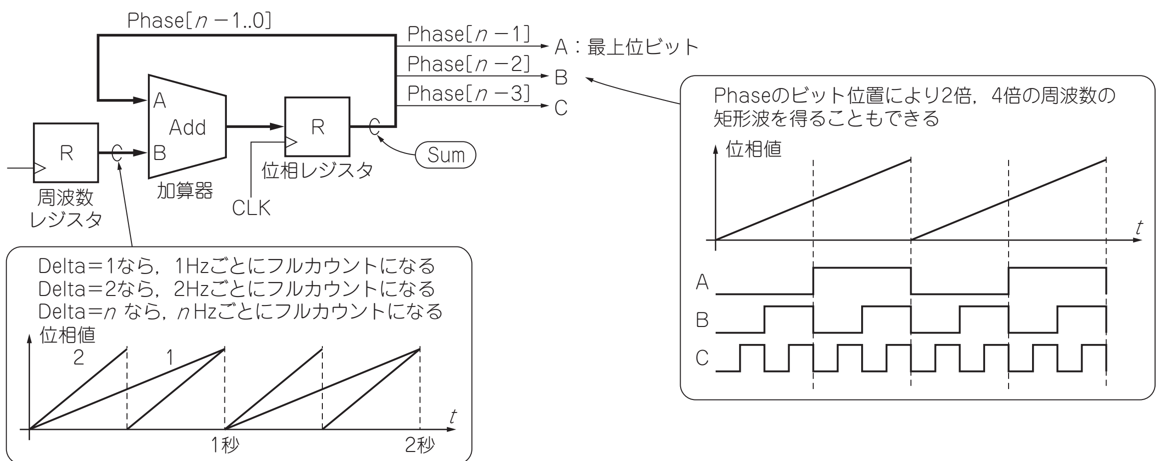


図3　可変周波数発振器の動作

は，ΣΔ型A-Dコンバータ，可変周波数発振器，シリアル通信(送受信)回路などを搭載しています(**図2**).

CPLDで作るから…開発環境の準備

● 入手先

写真1は心電計用ボードを流用した実験回路ですが，ボードに実装しているCPLDは，EPM240T100C5(アルテラ)ではなく，EPM570T100C5です．どちらのICも実装できるように設計しています.

今回，可変周波数発振器はDFL(Direct Frequency Loop)方式を用いていますが，Bluetooth経由で周波数を可変にすると，EPM240T100C5ではぎりぎり回路を入れることができませんでした．このディジタル回路設計はQuartus II ウェブ・エディションを使って行っています．このツールはアルテラ社のホームページから無償版をダウンロードして利用できます．URLは，`https://www.altera.co.jp/downloads/download-center.html`です．過去の無償バージョンもダウンロードできます.

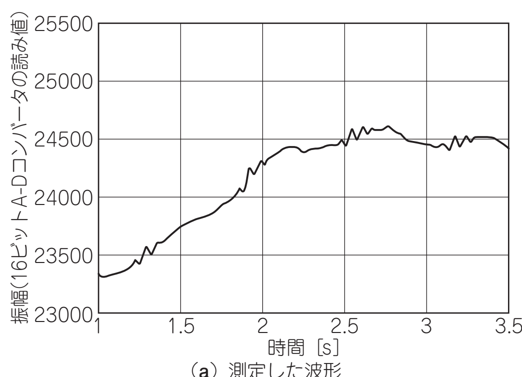
このツールには幾度も改良が加えられてはきているのですが，かえって不便になってしまった機能もあるため，このように過去のバージョンが利用できることは，ユーザにとってはありがたいものです.

● Verilog HDLやVHDLだけでなく回路図によるロジック設計も行える

Quartus II ウェブ・エディションでは，Verilog HDLやVHDLといったHDL(Hardware Design Language)を用いてディジタル回路設計を行うことができますが，筆者のように回路図ベースの設計も行うことができます.

コラム　心拍数を求めるプログラム

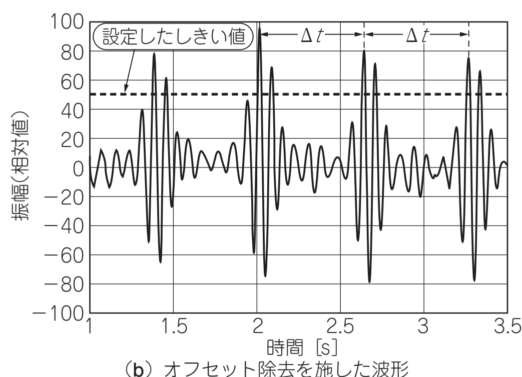
上田 智章



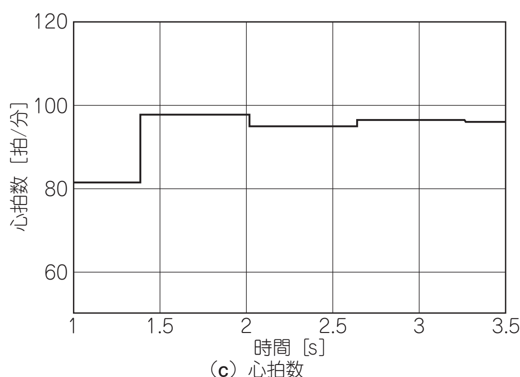
(a) 測定した波形

前回，静電容量型センサで胸部において測定した呼吸・心拍が混入した信号から心拍成分(心弾動)を分離する方法について説明しました．心拍検出用の矩形波関数を用いることで，**図A**(a)の呼吸に混入した心拍信号を**図A**(b)のように分離できるということを示しました．

最大ピーク値(正値)の半分に相当する50をしきい値とすると，これを超える短い周期の波が2回ずつ続きます．しかも第1波の方が，常に第2波より大きな値を持ちます．従って，心拍のタイミングを



(b) オフセット除去を施した波形



(c) 心拍数

図A　静電容量型センサでは心拍数も測定できる
しきい値を超えた第1波のピーク間の時間Δtを用いて心拍数 [拍/分] を求める．60/Δt

ソフトウェア制御周波数発振器の回路づくり

● 基本構成…加算器とレジスタ

可変周波数発振器を使えば，たった一つのクロックからシリアル通信で用いるボー・レートのクロック(送信用×1倍，受信用×4倍)，静電容量型センサの任意周波数の駆動信号(正論理/負論理)を生成できます．

可変周波数発振器は**図3**に示すように，一つの加算器と二つのレジスタを組み合わせて構成される累積加算回路になっています．累積加算値(Sum)を保持するレジスタを位相レジスタ，毎回加算する値(Delta)を保持するレジスタを周波数レジスタと呼びます．位相レジスタに供給されるシステム・クロックの周波数をCLK[Hz]とし，ビット幅をnビットとします．

仮にビット幅がn = 15ビットで，システム・クロックがCLK = 32768 = 2^{15}[Hz]とした場合，Delta = 1なら32768クロックごとにSumはフルカウントになるため，Sum[14]は1Hz周期で‘0’と‘1’を繰り返すことになります．Delta = 2であれば周期は2Hz，Delta = 3であれば周期は3Hzとなります．また，最上位ビットであるSum[14]が‘0’と‘1’を繰り返す間に，Sum[13]は‘0101’，Sum[12]は‘01010101’と位相レジスタのビット位置が1段低くなるにつれ，2倍の周波数になります．

● 作るもの1…シリアル通信用クロック生成回路

10MHzのシステム・クロックから115,200bpsのボー・レート・クロックを作る事例を紹介します．ビット幅を17ビットとした場合，Deltaを1秒間に10メガ回加算して，17ビットで桁あふれを起こす回数が115,200回であるので次式が得られます．

$$115200 = \frac{\mathrm{Delta} \times 10^7}{2^{17}}$$

$$\mathrm{Delta} = \frac{115200 \times 2^{17}}{10^7} = 1509.949$$

$$\fallingdotseq 1510$$

1510で駆動するときの周期は115203.9Hzとなるの

第4回　心拍数を取得する(2016年3月号)
第5回　リラックス心拍解析に挑戦(2016年4月号)
第6回　お風呂で非接触心拍解析(2016年5月号)

第1波のピークと定めることができます．第2波を誤検知しないためには，第1波検出後100～200ms間のデータを無視するようにすれば，後は心電図の処理と同じように取り扱えます．結果を**図A**(**c**)に，実際のコードを**リストA**に示します．

リストA　心拍数(*R*-*R*間隔)**を求める**

```
'***【R波トリガークラス】
Option Explicit

Public nQRS As Long
 ' リスト番号兼、格納ポインタ、格納個数
Private QRS() As Double
 ' 第1波ピーク時刻格納用の配列
Private QRSThreshold As Double ' しきい値
Private maxData As Double
 ' ピーク検出の為の作業用データ
Private maxTime As Double
 ' ピーク検出の為の作業用時刻
Private Flag As Long
 ' 0: ピーク探索ステップ  1: スキップフェーズ
Public HeartBeatRate As Double  ' 心拍数[拍/分]
Public TimeSpan As Double
 ' 第1波ピーク後、探索をスキップする時間[秒]

'***【Initialize】
Public Sub Initialize(Thd As Double, Span As Double)
   Flag = 0
   nQRS = 0
   ReDim QRS(1000)
   maxData = 0#
   maxTime = 0#
   QRSThreshold = Thd
   HeartBeatRate = 0#
   TimeSpan = Span
End Sub

'***【GetHeartBeatRate】
Public Function GetHeartBeatRate(t As Double,
                                D As Double) As Double
   Select Case Flag
      Case 0 'ピーク時刻を探索するフェーズ
         If (D > QRSThreshold) Then
            If (D >= maxData) Then
               maxData = D
               maxTime = t
            Else 'ピークを過ぎたのでさっきの値がピーク時刻だった.
               QRS(nQRS) = maxTime
               nQRS = nQRS + 1
               Flag = 1
               maxData = 0#
            '  maxTime = 0       コメントアウト
            End If
         End If

      Case 1 '0レベル以下までスキップするフェーズ
         If ((D < 0) And (t > (maxTime + TimeSpan)))
                                                Then
            Flag = 0
         End If
   End Select

   '【HeartBeatRate計算】
   If (nQRS > 1) Then
      HeartBeatRate = 60# / (QRS(nQRS - 1) -
                                    QRS(nQRS - 2))
   End If
   GetHeartBeatRate = HeartBeatRate
End Function
```


で，誤差は0.0033％となります．**図4**に実際の回路を示します．

● 作るもの2…発振回路

冒頭でも述べましたが筆者オリジナルの静電容量型センサの駆動回路では，人体のどの部分の生体組織誘電率の変化を測定するのかで，電極面積が変わります．そして，その電極面積に応じて共振周波数が大きく変化してしまうので，駆動周波数の方をソフトウェアにより最適な周波数に変化させてやる必要が生じます．

このためBluetooth経由でパソコン側から可変周波数発振器の駆動周波数を変更する値(PC_Delta)を設定変更する仕様が必須となります．また，可変周波数発振器の駆動出力はできる限り遅延のない差動出力である必要もあります．従って**図5**に示すような可変周波数発振器を設計しました．駆動周波数f_D[Hz]は次式で表せます．

$$f_D = \frac{\text{PC_Delta} \times 10^7}{2^{16}}$$

正論理と負論理の駆動信号に遅延が生じないように，Q[15]を反転した信号をQ[16]として，レジスタで保持しています．また，加算器のキャリ(桁上がり)出力もレジスタで保持しています．

S[17](DAout)は，PC_Deltaの値が高いほど，‘1’の密度(単位時間当たりの桁上がり回数)が高くなるので，抵抗とコンデンサだけの簡単なローパス・フィルタを通すだけで，パソコンからの設定値が正しく送信できたかどうかをチェックできます．

＊　＊　＊

http://www.neo-tech-lab.co.jp/VitalMonitorにさらに詳細な追加情報やサンプル・ソースコードを掲載します．ぜひご利用ください．

次回は具体的に，測定対象の筋肉の目標を定め，センサを設計，キャリブレーションを行う部分を説明します．

うえだ・ともあき

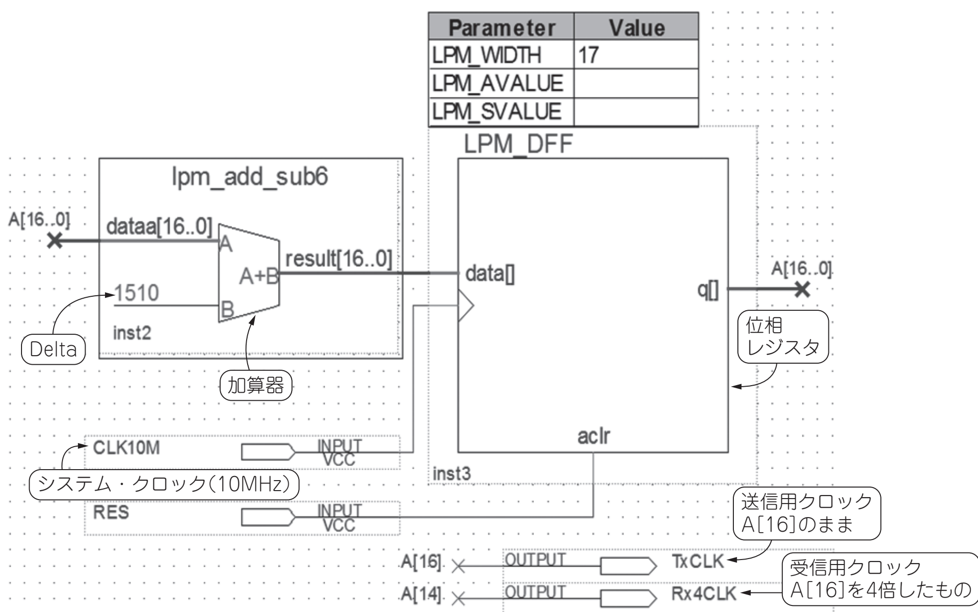


図4 可変周波数発振器のボー・レート・ジェネレータへの応用

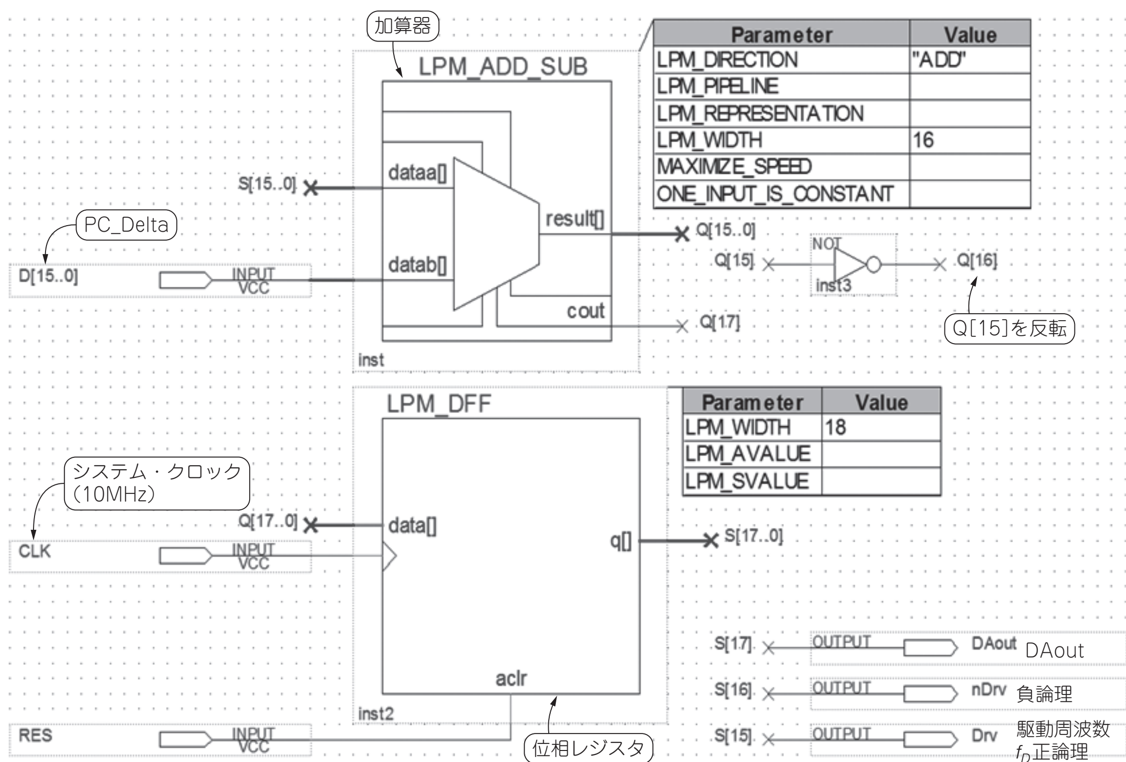


図5 静電容量型センサのための周波数駆動回路

パソコンからの指令値(PC_Delta)で周波数が変わる

ダウンロード・データあります

パケットづくりではじめるネットワーク入門

第15回 ポート開放機能を追加してホーム・サーバをインターネットに公開する

坂井 弘亮

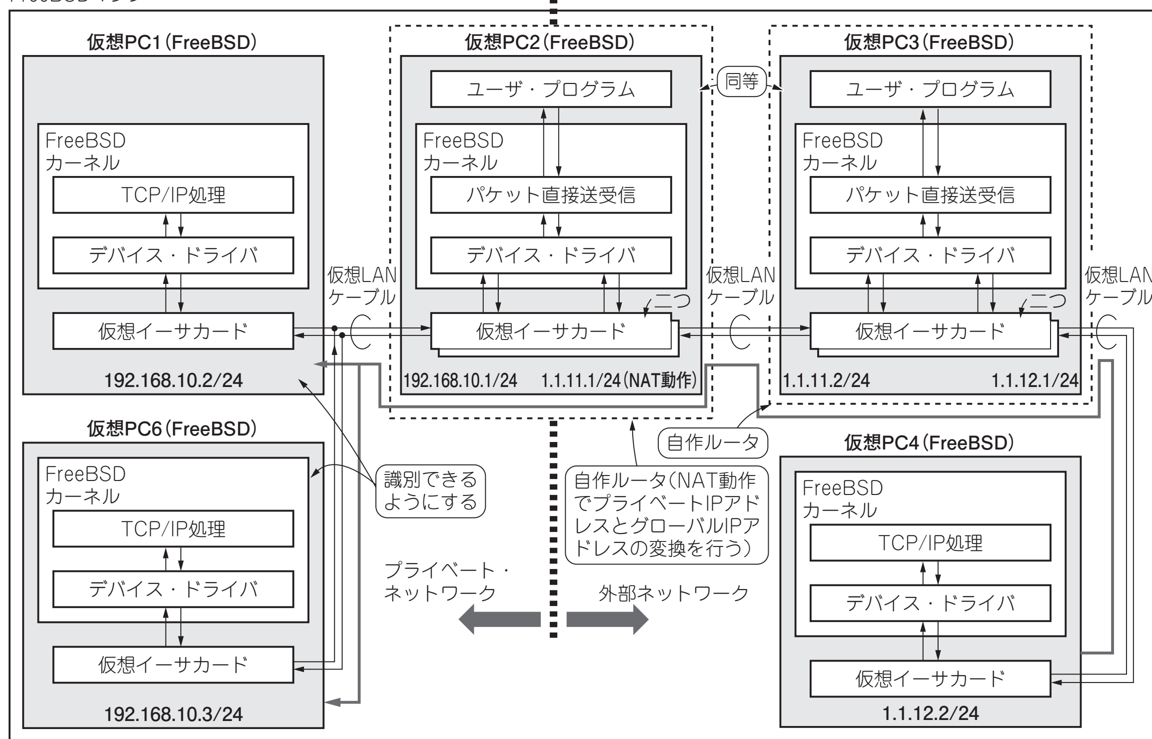


図1　今回やること…簡易ルータにポート開放の機能を追加することでプライベート・ネットワーク内のホーム・サーバをインターネットに公開する

● 今回行うこと…ホーム・サーバをインターネットに公開する

前回は簡易ルータのNAT(Network Address Translation, ここでは正確にはNAPT)機能をTCPとICMPに対応させました．これでホーム・ネットワークとインターネットを橋渡しするゲートウェイに利用できるルータになりました．いわゆる「ブロードバンド・ルータ」になったわけです．

今回はホーム・ネットワーク内のサーバをインターネットに公開するための「ポート開放」の機能を簡易ブロードバンド・ルータに追加します(**図1**)．

ホーム・サーバをインターネットに公開するには

● 実は今まで考えてきたのは内部から外部への通信

ここでは，プライベート側は家庭内ネットワークのようなローカル・ネットワーク，グローバル側はインターネットであり，サーバはインターネット上にあることを想定しています．

NATを利用することで，ネットワークをプライベート側とグローバル側に分離するとともに，プライベート側のノードはルータが持つグローバルIPアドレスだけを用いてグローバル側のノードと通信が可能

本連載のプログラムのソースコードは以下の筆者のホームページからダウンロードできます．ライセンスはKL-01というもので，組み込み機器などでも自由に利用できます．
http://kozos.jp/books/interface/ethernet2/

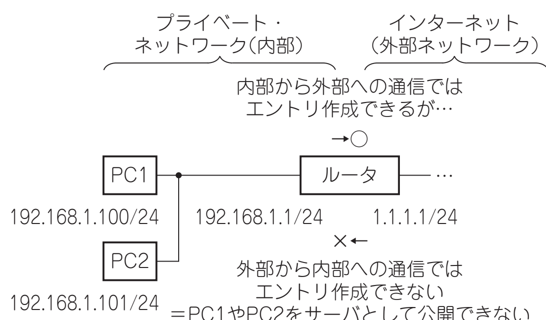


図2　通常のNAT動作だと外部（グローバル側）**から内部**（プライベート側）**への通信でNATのエントリは作成されない**

になります．

これは，家庭内ネットワーク上のPCからインターネット上のサーバに接続するだけならば，特に問題はありません．例えば家のPCからインターネット上のウェブ・サーバにブラウザで接続したり，そのためのDNSによる名前解決を，やはり家のPCからインターネット上のDNSサーバに問い合わせたりするような場合です．つまり「内部から外部への通信」です．

ここで，プライベート側にサーバを配置して，グローバル側にサービスを提供したくなったとします．例えば家庭内ネットワークに置いてあるPCをウェブ・サーバとしてインターネット公開するような場合です．これは「外部から内部への通信」になります．このとき，どのような問題点と解決策があるでしょうか．

● NATの問題点…外部のインターネットからプライベート・ネットワーク内部にあるサーバに通信を開始できない

NATのエントリが作成されるきっかけは，プライベート側(内部)からグローバル側(外部)への通信の開始です．

逆に，グローバル側(外部)からプライベート側(内部)への通信の開始を契機としてエントリを作成することはできません．

これは，プライベート側からの通信がない状態では，ルータはグローバル側で受信したパケットの宛て先IPアドレスを，どのプライベートIPアドレスに変換すればよいかが判断できないためです．例えば**図2**のようにプライベート・ネットワーク側に複数のノードがあった場合に，ルータは変換先をPC1にすべきかPC2にすべきかを，判断できません．

そして，これはプライベート側のサーバをグローバル側に公開する場合に問題となります．

サーバはクライアントからのリクエストを受け付け，サービスを返すものです．つまり動作のきっかけは，クライアント側からの通信の開始になります．これはエントリの作成のきっかけとは逆の方向です．これではNATのエントリの作成ができないことから，通信を開始することができないわけです．

● 解決策…特定ポートの転送先をあらかじめ決めておけば外部からもアクセスできる

これは通信用のエントリを固定であらかじめ作成しておくことで解決できます．つまり，通信の開始を契機としてエントリを自動作成するのではなく，ルータの特定のポートに対するアクセスをプライベート・ネットワーク上のどのノードに転送するのか，固定であらかじめ設定しておくわけです．

インターネット上のクライアントからは，ルータのグローバルIPアドレスと特定ポートに対して接続すればよいことになります．よってインターネット側からは，ルータが特定ポートでウェブ・サーバなどのサービスを行っているように見えます．が，実際にはNAT変換されてプライベート・ネットワーク内(内部側)のサーバに通信は転送されているわけです．

このような機能は「ポート開放」などと呼ばれます．他にも「ポート・フォワーディング」，「静的NAT」，「スタティックNAT」など，ルータによってさまざまな呼ばれ方をしているようです．

開放するポートの番号を変えることで，転送先のサーバを切り替えられます．例えばPC1とPC2の両方でウェブ・サーバ(HTTPのポート番号は80)を動作させる場合には，ルータのポート開放の設定では，10080番ポートへの通信はPC1に，20080番ポートへの通信はPC2に，というように振り分けることで両方のウェブ・サーバを公開できます．

自作簡易ルータのプログラム

ポート開放はNATのエントリを固定で設定することで実現できます．このためにはエントリの登録用のサービス関数を作成し，簡易ルータの起動時にそれを呼び出して固定でエントリを登録するようにします．

● エントリの追加用サービス関数

まずはサービス関数の作成です．**リスト1**(nat.c)は，前回(第14回)で作成した簡易ルータのNAT機能の部分(nat.c)に対して，NATのエントリ登録用のサービス関数を追加したものです．

リスト1ではエントリの登録のために，natif_add_entry_tcp()とnatif_add_entry_udp()の二つの関数を新たに追加してあります．これらはそれぞれTCP/UDPのエントリ用です．

エントリの追加のためにはnattable_add_entry()というサービス関数が既に実装されている

第1回　パケット送受信のライブラリを作成する（2015年8月号）
第2回　中継も速度測定も試せる！指定サイズ・パケット送信ライブラリを作る（2015年9月号）
第3回　抽象化しておけば超便利！バッファ付きパケット通信ライブラリを作る（2015年10月号）

リスト1　NATのエントリ登録用サービス関数(nat.c)

```
 :
int natif_add_entry_tcp(natif_t natif, int
             global_port, in_addr_t local_ipaddr,
                                  int local_port)

{
  struct natentry *entry;
  entry = nattable_add_entry(&natif->tcp,
       natif->ipaddr, local_ipaddr, local_port);

  entry->global.port = global_port;
  return 0;
}

int natif_add_entry_udp(natif_t natif, int
             global_port, in_addr_t local_ipaddr,
                                  int local_port)

{
  struct natentry *entry;
  entry = nattable_add_entry(&natif->udp,
       natif->ipaddr, local_ipaddr, local_port);

  entry->global.port = global_port;
  return 0;
}
```

リスト2　エントリの登録処理(router.c)

```
    :
099:int main(int argc, char *argv[])
100:{
    :
151:  argc--; argv++;
152:  /* 静的NATエントリを設定 */
153:  for (; argc > 4; argc -= 5, argv += 5) {
154:    for (ifp = iflist; ifp < &iflist[ifnum];
                                         ifp++) {
155:      if (!strcmp(pktif_get_name(ifp->pktif),
                                        argv[0]))
156:        break;
157:    }
158:    if (ifp != &iflist[ifnum]) {
159:      switch (argv[1][0]) {
160:      case 't':
161:        natif_add_entry_tcp(ifp->natif,
                                   atoi(argv[2]),
162:                 ntohl(inet_addr(argv[3])),
                                  atoi(argv[4]));
163:        break;
164:      case 'u':
165:        natif_add_entry_udp(ifp->natif,
                                   atoi(argv[2]),
166:                 ntohl(inet_addr(argv[3])),
                                  atoi(argv[4]));
167:        break;
168:      default:
169:        break;
170:      }
171:    }
172:  }
    :
```

ため，natif_add_entry_tcp()とnatif_add_entry_udp()の内部で行っているのは，TCPとUDPの管理テーブルに対してnattable_add_entry()を用いてエントリ登録するだけの処理です．

簡易ルータの起動時にはポート開放の設定に応じてこれらの関数を明示的に呼び出すことで，エントリを手動で登録できます．

● エントリの登録処理

リスト2(router.c)は，ポート開放のためのエントリの登録処理です．簡易ルータの起動時の処理部分に，起動時のコマンドライン引き数を参照してNATのエントリを登録する処理を追加しています．

今回追加したのは151～172行目です．その直前には，コマンドライン引き数を参照してインターフェースと経路の設定を行う既存の処理があります．それらについては連載の第8回と第9回を参照してください．

153～172行目ではコマンドライン引き数を順に参照して，エントリの登録を行います．

まず154～157行目のループで，引き数で指定されたインターフェースを検索します．さらに次の引き数の先頭文字を見て，「t」ならば161行目でTCPのエントリを，「u」ならば165行目でUDPのエントリを登録します．登録する内容は，順にグローバル側のポート番号，プライベート側のIPアドレス，プライベート側のポート番号です．

これにより，簡易ルータのグローバル側のインターフェースに対して，グローバル側のポート番号を宛て先として送られたパケットはエントリの検索にヒットし，NAT機能によってプライベート側のIPアドレスとポート番号に宛て先を変更されて転送されることになります．

このようにNATのエントリを手動で強制的に登録することが，ポート開放の機能になるわけです．よってNATにポート開放の機能を追加することは非常に簡単で，NAT機能を持つ多くのルータに実装されています．

ポート開放実験の準備

簡易ルータを動かして，ポート開放の動作を確認してみましょう．

● 簡易ルータのビルド

まず簡易ルータ(router.c)をビルドして，実行ファイルを作成します．ビルドの方法は前回と同様です．

コンパイルとリンクには，前回までに説明した以下のファイルがカレント・ディレクトリに必要です．これらについてはそれぞれの記事を参照してください．

①第3回パケット・ライブラリとパケット・バッファ・ライブラリ

- pktlib.h…ライブラリのヘッダ・ファイル
- pktbuf.h…パケット・バッファ関連のヘッダ・ファイル
- libpkt.a…ライブラリの本体

第4回　IPアドレスに対応するMACアドレスをゲット! ARP処理ライブラリを作る(2015年11月号)
第5回　ネットワークの診断に便利! ping応答ツールを作る(2015年12月号)
第6回　ネットワーク診断を自動で好きなように! ping送信ツールを作る(2016年1月号)

②第4回ARP処理ライブラリ

- arplib.h…ライブラリのヘッダ・ファイル
- libarp.a…ライブラリの本体

以下のようにコンパイル・リンクすることで，簡易ルータの実行ファイル(router)が作成されます．なおコンパイルはFreeBSDとCentOSで確認してあります．

```
% gcc -Wall router.c nat.c -o router -L. -lpkt -larp(ここまでを1行で入力)
```

● ネットワーク構成

ポート開放の動作確認は**図1**のようなネットワーク構成で行います．

図1は，以下のような構成を想定しています．

- PC1 ～ PC2 ～ PC6…家庭内ネットワーク(プライベート側)
  IPアドレスは192.168.X.X(プライベートIPアドレス)
- PC2…インターネットに接続するためのゲートウェイとなるルータ
- PC2 ～ PC4…インターネット(グローバル側)
  IPアドレスは1.1.X.X(グローバルIPアドレス)

インターネットであるグローバル側は，グローバルIPアドレスに例として1.1.X.XというIPアドレスを利用しています．

PC2とPC3ではOSのルーティング機能は動作させず，今回作成した簡易ルータを動作させます．さらにPC2のPC3側のインターフェースでNATを動作させ，プライベートIPアドレスとグローバルIPアドレスの変換を行います．この構成でPC4からPC1への通信と，PC4からPC6への通信が可能なことを確認します．つまり，インターネット上にあるクライアントから，家庭内ネットワーク上にある二つのサーバに対して通信を行うわけです．

なおPCを実際に5台用意することは面倒であるため，筆者はVM(Virtual Machine；仮想マシン)の環境にて検証しています．

● ノードの設定

まずはPCを**図1**のように接続します．

さらにPC1，PC6，PC4にはIPアドレスを設定し，経路を登録しておきます．本稿での実験にはFreeBSDを利用していますが，CentOSなどでも同じように実施できます．以下はFreeBSDの場合ですが，Linuxではインターフェースにeth0などを指定します．

(PC1)

```
# ifconfig em1 192.168.10.2/24
# route add 1.1.0.0/16 192.168.10.1
```

(PC6)

```
# ifconfig em1 192.168.10.3/24
# route add 1.1.0.0/16 192.168.10.1
```

(PC4)

```
# ifconfig em1 1.1.12.2/24
# route add 1.1.0.0/16 1.1.12.1
```

● 簡易ルータの準備

PC2とPC3では簡易ルータを動作させるために，インターフェースをUPしておきます．さらにFreeBSDの場合には，BPFを読み書き可能にしておきます．

(PC2，PC3)

```
# ifconfig em1 up
# ifconfig em2 up
# chmod 666 /dev/bpf (FreeBSDの場合)
```

ポート開放の動作確認

■ 簡易ルータの起動

PC2とPC3で簡易ルータを動作させて，ポート開放によりグローバル側からプライベート側への通信が行えることを確認してみましょう．

● 起動方法

簡易ルータの実行時のコマンドライン引き数には，インターフェース情報，経路情報，ポート開放情報を「--」で区切って指定します．インターフェース情報と経路情報の指定方法は，前回までと同様です．ポート開放情報には，以下を順に指定します．

- インターフェース名
- TCP/UDPの指定
- グローバル側のポート番号
- プライベート側のIPアドレス
- プライベート側のポート番号

● 起動例

簡易ルータには，前回と同じように以下の経路を登録します．これはPC4と通信を行うためのものです．

- (PC2) 1.1.0.0/16 → 1.1.11.2

リスト3はFreeBSDでの起動例です．Linux環境の場合はスーパユーザで実行し，さらにインターフェースにはやはりeth0などを指定します．

PC2では以下の二つのポート開放情報を指定しています．

- em2 udp 10000 192.168.10.2 7
- em2 udp 10001 192.168.10.3 7

つまり，PC2のグローバル側でUDPの宛て先ポート番号が10000となっているパケットを受信すると，

それはプライベート側の192.168.10.2に，宛て先ポート番号を7に変換して転送されます．これはPC1が受信することになります．

また同じように，宛て先ポート番号が10001となっているパケットは，プライベート側の192.168.10.3に，やはり宛て先ポート番号を7に変換して転送されます．これはPC6が受信することになります．

なおこの「7」は，次に説明するechoサービスのポート番号です．

● echoサービスの起動

PC1，PC6では次のようにして，UDPのechoサーバを起動しておきます． echoサービスは通信を受けると送られてきた内容をそのまま返すだけの単純なサービスで，TCPのコネクションの接続性やUDPのパケット応答の確認に利用できます．サービスとしてはTCPとUDPの両方がありますが，今回はUDPで利用します．サービスのポート番号は，TCP/UDPともに「7」です．

FreeBSDでechoサービスを有効にするには，/etc/inetd.confで以下の行を有効にします．

(PC1，PC6)

```
echo      dgram     udp       wait
root      internal
```

さらにスーパユーザで以下を実行して，inetdをリスタートします．

(PC1，PC6)

```
# service inetd onestart (inetdが未起動の場合)
# service inetd restart (inetdが既に起動している場合)
```

この状態でPC1とPC6のUDPポート7に対してパケットを送信すると，応答が返されるはずです．

● パケット・キャプチャの起動

PC2のインターフェース上で，確認用にパケット・キャプチャを開始しておきます．

キャプチャにはpkttoolsというフリー・ソフトウェアを利用します． pkttoolsについては以下を参照してください．今回は現時点の最新版であるpkttools-1.14を利用します．

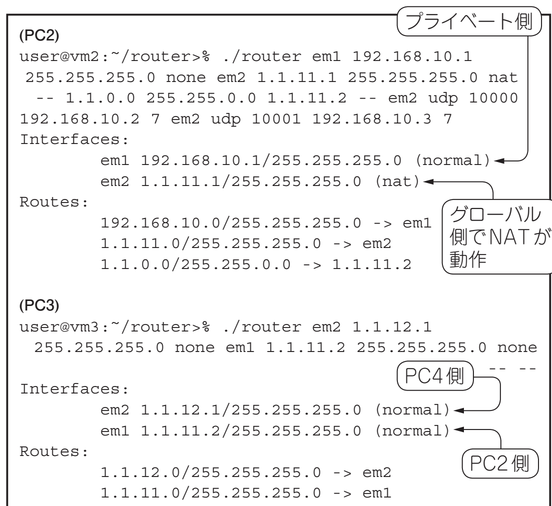
```
http://kozos.jp/software/pkttools.html
```

PC2で以下を実行して，キャプチャを開始します．キャプチャするインターフェースはem2（プライベート側）です．

(PC2)

```
user@vm2:~/pkttools-1.14>% ./pkt-recv -i em1 | ./pkt-analyze -ll 3 -lv 2 (ここまでを1行で入力)
```

リスト3　FreeBSDでの起動時



```
(PC2)
user@vm2:~/router>% ./router em1 192.168.10.1
 255.255.255.0 none em2 1.1.11.1 255.255.255.0 nat
  -- 1.1.0.0 255.255.0.0 1.1.11.2 -- em2 udp 10000
192.168.10.2 7 em2 udp 10001 192.168.10.3 7
Interfaces:
        em1 192.168.10.1/255.255.255.0 (normal)
        em2 1.1.11.1/255.255.255.0 (nat)
Routes:
        192.168.10.0/255.255.255.0 -> em1
        1.1.11.0/255.255.255.0 -> em2
        1.1.0.0/255.255.0.0 -> 1.1.11.2

(PC3)
user@vm3:~/router>% ./router em2 1.1.12.1
 255.255.255.0 none em1 1.1.11.2 255.255.255.0 none
                                              -- --
Interfaces:
        em2 1.1.12.1/255.255.255.0 (normal)
        em1 1.1.11.2/255.255.255.0 (normal)
Routes:
        1.1.12.0/255.255.255.0 -> em2
        1.1.11.0/255.255.255.0 -> em1
```


■ 通信実験

● UDPポート10000への通信

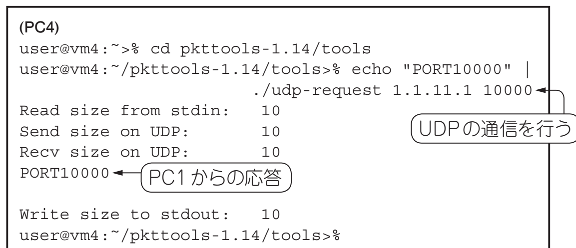
この状態でPC4からPC2のUDPポート10000に対して，UDPの通信を行ってみます（**リスト4**）．これにはpkttoolsに付属するudp-requestというコマンドが利用できます．udp-requestは指定したIPアドレスとポート番号に対してUDPのパケットを送信し，さらに応答を待ち受けます．

udp-requestに指定する宛て先IPアドレスが，PC2の1.1.11.1になっている点に注目してください．つまりPC4から見ると，通信相手はPC2に見えるわけです．しかし実際に通信を行う相手はPC1です．パケットがPC2上で期待通りにNAT変換されPC1上のechoサービスまで到達すれば，送信した文字列がそのまま応答として返ってくることになります．

応答が無事に返ってきており，UDPの通信ができているようです．**リスト5**は簡易ルータのログです．

一つ目はPC4から送信されたUDPのパケットです．宛て先IPアドレスは1.1.11.1（PC2）で，送信元IPアドレスは 1.1.12.2（PC4）です．そして宛て先IPアドレスは，192.168.10.2（PC1）に変換されています．

リスト4　UDPの通信（PC4からPC2のUDPポート10000へ）



```
(PC4)
user@vm4:~>% cd pkttools-1.14/tools
user@vm4:~/pkttools-1.14/tools>% echo "PORT10000" |
                              ./udp-request 1.1.11.1 10000
Read size from stdin:    10
Send size on UDP:        10
Recv size on UDP:        10
PORT10000

Write size to stdout:    10
user@vm4:~/pkttools-1.14/tools>%
```

リスト5　簡易ルータ(PC2)のログ

```
(PC2)
RECV IP Packet em2 1.1.12.2         -> 1.1.11.1
NAT(before) 1.1.12.2         -> 1.1.11.1
NAT(after)  1.1.12.2         -> 192.168.10.2
SEND IP Packet em1 192.168.10.2
                  一つ目のパケット(PC4からの送信)
RECV IP Packet em1 192.168.10.2    -> 1.1.12.2
NAT(before) 192.168.10.2    -> 1.1.12.2
NAT(after)  1.1.11.1         -> 1.1.12.2
SEND IP Packet em2 1.1.11.2
                  二つ目のパケット(PC1からの応答)
⋮
```

リスト6　PC2(プライベート側)インターフェースでのパケットをキャプチャ

```
(PC2)
user@vm2:~/pkttools-1.14>% ./pkt-recv -i em1 |
                          ./pkt-analyze -ll 3 -lv 2
⋮
-- 3 --                     (1)グローバル側からの受信
IP      src/dst addr    : 1.1.12.2 / 192.168.10.2
UDP     src/dst port    : 57287 / 7
-- 4 --
IP      src/dst addr    : 192.168.10.2 / 1.1.12.2
UDP     src/dst port    : 7 / 57287
                          (2)プライベート側からの受信
```

二つ目はPC1からechoサービスにより返された応答パケットです．送信元が192.168.10.2(PC1)で宛て先は1.1.12.2 (PC4)になっていますが，やはりNATにより送信元は1.1.11.1(PC2)に変換されています．

よってPC4から見えるのは1.1.11.1というPC2のIPアドレスだけで，192.168.10.2というPC1のプライベートIPアドレスは見えないことになります．PC4からはルータであるPC2と通信しているように見えます．

リスト6はPC2のプライベート側のインターフェースでのパケットのキャプチャ結果です．

まず**リスト6**の(1)では，宛て先IPアドレスが192.168.10.2 のパケットが見えています．これはPC4から送信され，PC2でNAT変換されてPC1に転送されたパケットです．宛て先ポート番号は10000だったはずですが，7に変換されています．

次に**リスト6**の(2)では，送信元IPアドレスが192.168.10.2 のパケットが見えています．これは**リスト6**の(1)に対する，PC1の応答パケットです．PC1からの応答であるため，送信元IPアドレス(192.168.10.2)と送信元ポート番号(＝7)は，ともにプライベート側のものになっています．

よってIPアドレスとポート番号がNAT変換され，プライベート・ネットワーク内ではプライベート側のIPアドレスとポート番号を利用した通信が行われていることが，パケット上からも確認できます．

リスト7　UDPの通信ログ(PC4→PC2のUDPポート10001)

```
(PC4)
user@vm4:~/pkttools-1.14/tools>% echo "PORT10001" |
                      ./udp-request 1.1.11.1 10001
Read size from stdin:   10            UDPの通信を行う
Send size on UDP:       10
Recv size on UDP:       10
PORT10001 ← PC6からの応答

Write size to stdout:   10
user@vm4:~/pkttools-1.14/tools>%
```

リスト8　簡易ルータ(PC2)の通信ログ
ポート番号の切り替えにより通信先のサーバを選択できる

```
(PC2)
RECV IP Packet em2 1.1.12.2         -> 1.1.11.1
NAT(before) 1.1.12.2         -> 1.1.11.1
NAT(after)  1.1.12.2         -> 192.168.10.3
SEND IP Packet em1 192.168.10.3
                  一つ目のパケット(PC4からの送信)
RECV IP Packet em1 192.168.10.3    -> 1.1.12.2
NAT(before) 192.168.10.3    -> 1.1.12.2
NAT(after)  1.1.11.1         -> 1.1.12.2
SEND IP Packet em2 1.1.11.2
                  二つ目のパケット(PC6からの応答)
⋮
```

リスト9　PC2(プライベート側)のキャプチャ結果

```
(PC2:プライベート側)
user@vm2:~/pkttools-1.14>% ./pkt-recv -i em1 |
                          ./pkt-analyze -ll 3 -lv 2
⋮
-- 7 --                     (1)グローバル側からの受信
IP      src/dst addr    : 1.1.12.2 / 192.168.10.3
UDP     src/dst port    : 26335 / 7
-- 8 --
IP      src/dst addr    : 192.168.10.3 / 1.1.12.2
UDP     src/dst port    : 7 / 26335
                          (2)プライベート側からの受信
```

● UDPポート10001への通信

同じように，PC4からPC2のUDPポート10001に対して，UDPの通信を行ってみます(**リスト7**)．

無事にUDPの通信ができているようです．**リスト8**は簡易ルータのログです．

ログの流れはPC1のときと同様ですが，今度は宛て先IPアドレスが192.168.10.3に変換されています．さらに応答も送信元IPアドレスは192.168.10.3になっています．つまり通信の相手はPC1ではなくPC6になっています．ポート開放ではこのようにポート番号を切り替えることで，通信先のサーバを選択できます．

リスト9はキャプチャ結果です．

リスト9の(1)の宛て先IPアドレスと，それへの応答である**リスト9**の(2)の送信元IPアドレスは，192.168.10.3になっています．よって通信の相手は，やはりPC1ではなくPC6になっていることが，実際の通信パケットからも分かります．

さかい・ひろあき

第13回　プライベート・ネットワークOK! UDP対応NAPTを作る(2016年8月号)
第14回　自作ルータのNATをTCP/ICMPに対応させる(2016年9月号)

Windows/Mac/Linux対応でI/Oもサクッ！

オープンソースのブロック型言語 Pure Dataではじめる サウンド信号処理

青木 直史，藍 圭介

第11回 C言語オリジナル・ブロックの作り方

● 今回の目標…C言語でオリジナル・ブロックを作ってみる

Pure Dataには，あらかじめ用意されているブロックだけでなく，C言語を使って独自のブロックを作成し，オリジナル・ブロックとして利用できるという拡張性があります．Pure Dataを使いこなすための奥義の一つとして，今回は，こうしたオリジナル・ブロックの作り方について説明し，実際に簡単なブロックを作成してみることにします．

● Pure DataでCプログラムが使えると守備範囲が広がる

ブロックを組み合わせるという直感的な操作で，さまざまなサウンド処理を手軽に試すことができるのがPure Dataの最大の特徴です．何はともあれ，まずは音を鳴らしてみたい初心者にとって，Pure Dataは便利なプログラミング環境といえるでしょう．

しかし，ブロックの組み合わせでプログラムを作ることは，一見すると簡単そうでも，それなりの慣れを必要とします．複雑な処理の場合，ブロックやケーブルがパッチ・ウィンドウを埋めつくしてしまい，文字通り，可読性の良くないスパゲティ・プログラムになってしまう恐れも少なくありません．ここに，Pure Dataのスケーラビリティの問題があり，本格的な利用に二の足を踏んでしまう原因があるように思います．

こうした問題の一つの解決策になるのが，オリジナル・ブロックの導入です．実は，Pure Dataには，あらかじめ用意されているブロックだけでなく，C言語を使って独自のブロックを作成し，オリジナル・ブロックとして利用できるという拡張性があります．ブロックの組み合わせに頭を悩ませるような場合でも，こうしたプログラミングの仕組みが用意されていることを知っておけば，Pure Dataの守備範囲を広げるためのノウハウとして大いに役立つのではないでしょうか．

その1…Windowsを使用する場合

● 手順1-1…Cコンパイラのインストール

オリジナル・ブロックを作成するにはC言語コンパイラが必要になります．Windowsの場合，C言語コンパイラとして最も一般的なのはVisual C++でしょう．ここでは，フリーのものとしてCommunity版を利用します．Community版は次のウェブ・サイトから「Download Community Free」をクリックすることでインストールできます．

```
https://www.visualstudio.com/en-us/
downloads/download-visual-studio-
vs.aspx
```

Visual Studioのインストールが完了したら，Visual Studioを起動し，「File」→「New」→「Project」を選択します．プロジェクトの種類として「Visual C++」を選択し，「Install Visual C++ 2015 Tools for Windows Desktop」を選択してVisual C++のツールをインストールします．

● 手順1-2…環境変数の設定

オリジナル・ブロックの作成は，コマンド・プロンプトを利用して作業するのが便利です．そのためのセットアップとして環境変数を設定します．「スタート」→「コンピュータ」を選択し，右クリックして「プロパティ」を選択します．続いて，「システムの詳細設定」から「環境変数」を選択し，ユーザ変数の中にあるPathを選択した後，「編集」を選択します．**図1**に示すように，Pathに「`;C:¥Program Files`

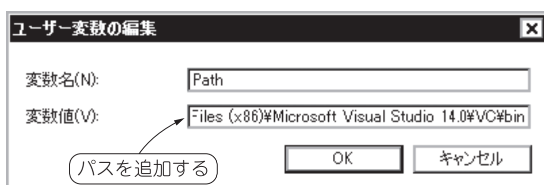


図1　環境変数の設定

リスト1　[hello] ブロックのプログラム(hello.c)

```
#include "m_pd.h"

t_class *hello_class;

typedef struct hello
{
  t_object x_obj;
} t_hello;

void hello_bang()   ← bangメッセージを受け取ったときの処理
{
  post("Hello, world!");
}

void *hello_new(void)   ← パッチ・ウィンドウにブロックを生成したときの処理
{
  t_hello *x = (t_hello *)pd_new(hello_class);
  return (void *)x;
}

void hello_setup(void)   ← Pure Dataを起動したときの処理
{
  hello_class = class_new(gensym("hello"),
  (t_newmethod)hello_new, 0, sizeof(t_hello), 0, 0);
  class_addbang(hello_class, hello_bang);
}
```

リスト2　Windowsで [hello] ブロックをコンパイルするためのmakefile

```
VC="C:¥Program Files (x86)¥Microsoft Visual Studio
                                          14.0¥VC"
VCINC1="C:¥Program Files (x86)¥Windows Kits¥10¥
                           Include¥10.0.10150.0¥ucrt"
VCLIB1="C:¥Program Files (x86)¥Windows Kits¥10¥Lib¥
                             10.0.10150.0¥ucrt¥x86"
VCLIB2="C:¥Program Files (x86)¥Windows
                          Kits¥8.1¥Lib¥winv6.3¥um¥x86"
    (↑ VC～VCLIB2: 環境に合わせて修正する)

all: hello.dll   ← [hello] ブロックをコンパイルするための設定

SUFFIXES: .obj .dll

CFLAGS = /W3 /wd4091 /DNT /DPD /nologo
INCDIR = /I. /I$(VC)¥include /I$(VCINC1)
LDDIR = /LIBPATH:$(VC)¥lib /LIBPATH:$(VCLIB1) /
                                      LIBPATH:$(VCLIB2)
LIBS = pd.lib

c.dll:
  cl $(CFLAGS) $(INCDIR) /c $*.c
  link /dll $(LDDIR) /export:$*_setup $*.obj $(LIBS)
```

(x86)¥Microsoft Visual Studio 14.0¥VC¥bin」を追加し,「OK」をクリックします.

● 手順1-3…オリジナル・ブロックのCプログラム作成

実際に,簡単なブロックを作成してみましょう.まず,作業用フォルダとして,Cドライブにhelloという名前のフォルダを作成し,その中に,**リスト1**のプログラムを保存します.プログラムの名前はhello.cとしてください.なお,このプログラムの名前がオリジナル・ブロックの名前になります.

● 手順1-4…ヘッダ・ファイルとライブラリのコピー

このプログラムをコンパイルするには,ヘッダ・ファイルとライブラリが必要になります.これらのファイルは,次のウェブ・サイトから「Get Pure Data for Windows (8 MB) Windows ZIP archive」をクリックすることでダウンロードできます.

```
https://puredata.info/downloads/pure-data
```

ダウンロードした圧縮ファイルを開き,srcフォルダの中にあるm_pd.hと,binフォルダの中にあるpd.libを,それぞれ作業用フォルダにコピーします.

● 手順1-5…コンパイルの実行

コンパイルはnmakeコマンドによって行います.作業用フォルダの中に,**リスト2**のmakefileを保存した後,コマンド・プロンプトに次のようにコマンドを入力し,実行します.なお,makefileの中に記述されているパスは,使用している環境に合わせて修正が必要になる場合があります.

```
> cd c:¥hello
> nmake
```

コンパイルが完了すると,作業用フォルダにhello.dllが作成されます.このファイルがオリジナル・ブロックにほかなりません.

● 手順1-6…Pure Dataプログラムの作成

実際に,このオリジナル・ブロックを使って,Pure Dataのプログラムを作ってみることにしましょう.**図2**に示すように,パッチ・ウィンドウにブロックを配置し,ブロックの名前としてhelloと入力すると,[hello] ブロックが作成されるはずです.

● 手順1-7…Pure Dataプログラムの実行

プログラムを実行してみましょう.[bang] ブロックをクリックするたびに,Pdウィンドウに「Hello, world!」というメッセージが表示されることがお分かりいただけるでしょうか.

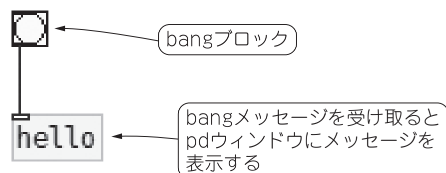


図2　[hello] ブロックを使ったPure Dataのプログラム(fig2.pd)

その2…Mac OSを使用する場合

● 手順2-1…C言語コンパイラのインストール

Mac OSの場合，C言語コンパイラとして最も一般的なのはXcodeでしょう．XcodeはApp Storeからインストールできます．

● 手順2-2…オリジナル・ブロックのプログラム作成

Windowsを使用するときの手順1-3と同様，作業用フォルダにhello.cを保存します．

● 手順2-3…ヘッダ・ファイルのコピー

オリジナル・ブロックのプログラムをコンパイルする場合，Windowsではm_pd.hとpd.libを作業用フォルダにコピーする必要がありますが，Mac OSではm_pd.hだけで十分です．次のウェブ・サイトから「Get Pure Data for All platforms (Source code)(2 MB) source tarball」をダウンロードし，srcフォルダの中にあるm_pd.hを作業用フォルダにコピーします．

```
https://puredata.info/downloads/pure-data
```

● 手順2-4…コンパイルの実行

コンパイルはmakeコマンドによって行います．作業用フォルダの中に，**リスト3**のmakefileを保存した後，ターミナルに次のようにコマンドを入力し，実行します．なお，makefileの中に記述されているオプションは，使用している環境に合わせて修正が必要になる場合があります．Pure Dataのバージョンが64ビット版の場合はx86_64になっているところを，32ビット版の場合はx86_32に修正します．

```
> cd hello
> make
```

コンパイルが完了すると，作業用フォルダにhello.pd_darwinが作成されます．このファイルがオリジナル・ブロックにほかなりません．

● 手順2-5…Pure Dataプログラムの作成

Windowsを使用するときの手順1-6と同じ方法で行います．

● 手順2-6…Pure Dataプログラムの実行

Windowsを使用するときの手順1-7と同じ方法で行います．

リスト3　Macで[hello]ブロックをコンパイルするためのmakefile

```
all: hello.pd_darwin

SUFFIXES: .pd_darwin

DARWINCFLAGS = -DPD -O2 -Wall -W -Wshadow
                                   -Wstrict-prototypes ¥
-Wno-unused -Wno-parentheses -Wno-switch -arch i386
                                          -arch x86_64

c.pd_darwin:
  cc $(DARWINCFLAGS) $(LINUXINCLUDE) -o $*.o -c $*.c
  cc -bundle -undefined suppress -arch i386 -arch
                                             x86_64 ¥
  -flat_namespace -o $*.pd_darwin $*.o

clean: ; rm -f *.pd_darwin *.o
```

[hello]ブロックをコンパイルするための設定

Pure Dataのバージョンが32ビット版のときはx86_32に修正する

オリジナル・ブロックを作成してみて分かるPure Dataの動作

あらためて，**リスト1**のプログラムを眺めてみましょう．オリジナル・ブロックのプログラムは，オブジェクト指向のコンセプトに従って記述されており，Pure Dataを起動したときの処理や，パッチ・ウィンドウにブロックを生成したときの処理など，それぞれのイベントによって呼び出される関数を記述したものになっています．

例えば，このプログラムでは，Pure Dataを起動したときの処理としてhello_setup関数，パッチ・ウィンドウにブロックを生成したときの処理として*hello_new関数が記述されています．hello_bang関数は，[hello]ブロックがbangメッセージを受け取ったときに呼び出される関数であり，このプログラムでは，この関数が[hello]ブロックの動作そのものに対応しています．

＊　＊　＊

今回は，簡単な例として，bangメッセージを受け取って動作するブロックを作成してみました．もちろん，Pure Dataには，こうしたコントロール・レートで動作するブロックだけでなく，音データをリアルタイムに処理するため，オーディオ・レートで動作するブロックもあります．

次回は，こうしたチルダ・ブロックをターゲットとして，オリジナル・ブロックの作り方についてさらに詳しく説明します．

あおき・なおふみ，あい けいすけ

ダウンロード・データあります

数mをリアルタイムに！クルマに使われる高信頼性バス！

制御＆監視向け！小型ネットワークCAN通信入門

最終回
第9回　基本的な送受信プログラムを作る

高島 光

表1　送受信するフレームの仕様

フレーム種別	ID種別	CANID	DLC	データ内容	送受信方向	送信タイミング
データ・フレーム	標準ID	0x001	任意	任意	受信	任意
		0x008				
		0x110～0x11F				
		0x220～0x22F				
		0x330～0x33F				
		受信したID+1	受信したDLCと同一	受信したデータを反転	送信	受信に対して即時に応答
		0x555	2	タイマ1カウンタ		1000ms周期

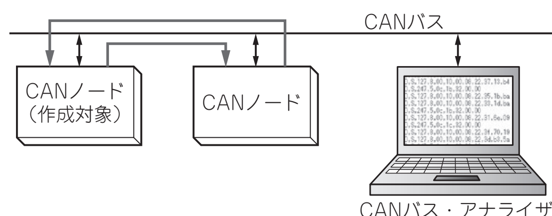


図1　二つのノード間で基本的な送受信を行うプログラムを作る

今回は，実際にCAN通信機器に使用するソフトウェアを作成します．

CAN通信は自由度が高いため，CANバス上に流れるフレームのルールを決めて通信を行う必要があります．この通信ルールを守るためには，OSEK/VDXやAUTOSARなどの標準化された仕様に基づいて作成されたミドルウェアを使用するのが一般的です．

今回はCANコントローラを制御するドライバ・ソフトウェアに焦点を当てて，実際にCANコントローラをどのように制御するのかを紹介します．

図1のように，バス上に二つのノードを接続し，**表1**のようなフレームを送受信するプログラムを作成します．

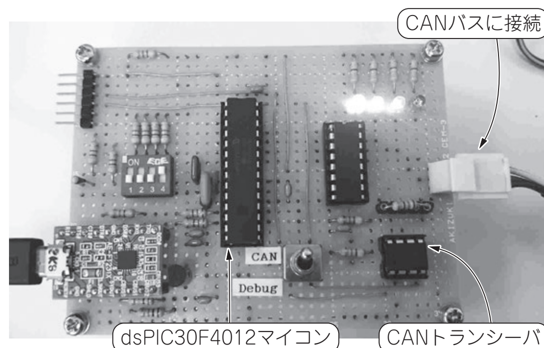


写真1　通信実験に使ったCANノード
第7回（2016年7月号）で作成したCAN通信実験回路．同様のキットを頒布予定あり．本誌ウェブ・サイト（http://interface.cqpub.co.jp/）などで紹介予定

開発環境の準備

● ハードウェア…自作CAN通信実験回路

CAN通信を行うソフトウェアを作成するにあたり，CANコントローラ（CANモジュール）を搭載したマイコンを使用するのが最も簡単です．現在，流通しているCANコントローラ搭載マイコンのほとんどは，ISO 11898-1仕様に準拠しており，CANプロトコル部分は，ハードウェアが自動的に実現してくれます．これにより，アービトレーション判定やエラー・ハンドリングなどの複雑な制御を意識することなく，アプリケーションで使用するデータ部分だけを意識して通信を行うことが可能となっています．

今回は，第7回（2016年8月号）で作成したCAN通信実験回路を使用します（**写真1**）．CAN通信実験回路には，CANコントローラ搭載のマイコンdsPIC30F4012が搭載されており，基本的な通信用プログラムを作成するのに適しています．

● ソフトウェア開発環境…無償で使える MPLAB X IDE&XC16コンパイラ

PICマイコンのソフトウェア開発環境として，マイクロチップの統合開発環境MPLAB X IDE(1)と，MPLAB XCコンパイラ(2)を使用します．評価版が無償で公開されているので気軽に利用できます．いずれもマイクロチップのウェブ・ページからダウンロードできます．本稿で使用するバージョンは，MPLAB X IDE v3.15とXC16 v1.25です．

使い方は，ユーザ・ガイド(3)が参考になります．デバイスの選択では，ターゲット・マイコンのdsPIC30F4012を選択します．

● デバッグ＆書き込み器…定番PICkit3

デバッグには，インサーキット・デバッガ/プログラマのPICkit 3(4)を使用します．MPLAB X IDE上でPICkit 3を使用してデバッグができます．MPLAB X IDEインストール時に，使用しているPC環境に合わせたUSBドライバがインストールされていることを確認してください．

● 送受信を確認するために…バス・アナライザ

作成したプログラムが正常にCANフレームの送受信を行えていることを確認するために，オシロスコープやバス・アナライザを準備します．

CANフレームの波形を確認するために，CANバス解析機能を搭載したオシロスコープが便利です．

また，各社から発売されているバス・アナライザでは，CANバス上のフレームをモニタするだけでなく，通信相手となるノードをシミュレーションする機能を持つものもあります．

dsPICマイコン内蔵 CANコントローラの設定

CANコントローラを使用して，CAN通信を行うまでの手順を**図2**に示します．

今回のプログラム作成では，CANコントローラ制御用のドライバ関数として，**表2**の関数を作成します．

● ステップ1：CANトランシーバを有効にする

マイコンに搭載されたCANコントローラは，マイコンのI/Oポートを使用して，送信信号の出力，受信信号の入力を行います．実際のCANバスに接続するためには，マイコンの入出力ポートをCANトランシーバに接続し，物理層で規定された信号に変換する必要があります．CAN通信実験回路では，一般的な構造を持つMCP2551（マイクロチップ）を使用しています（**図3**）．

CANトランシーバを制御するためには，マイコンの入出力ポートの設定と，CANトランシーバのモード設定が必要です．

▶①マイコンの入出力ポートの設定

通常のマイコンは，汎用入出力ポートをCAN通信で使用します．このため，CAN送受信で使用するポートの機能，入出力を設定します．

▶②CANトランシーバのモード設定

CANトランシーバは，低消費電力モードをサポートしています．CAN通信を開始するためには，マイコンから信号を入力するだけでなく，CANトランシーバを起動する必要があります．MCP2551では，Rs端子を“H”に設定するとスリープ・モード，“L”に設定するとトランシーバが動作します．

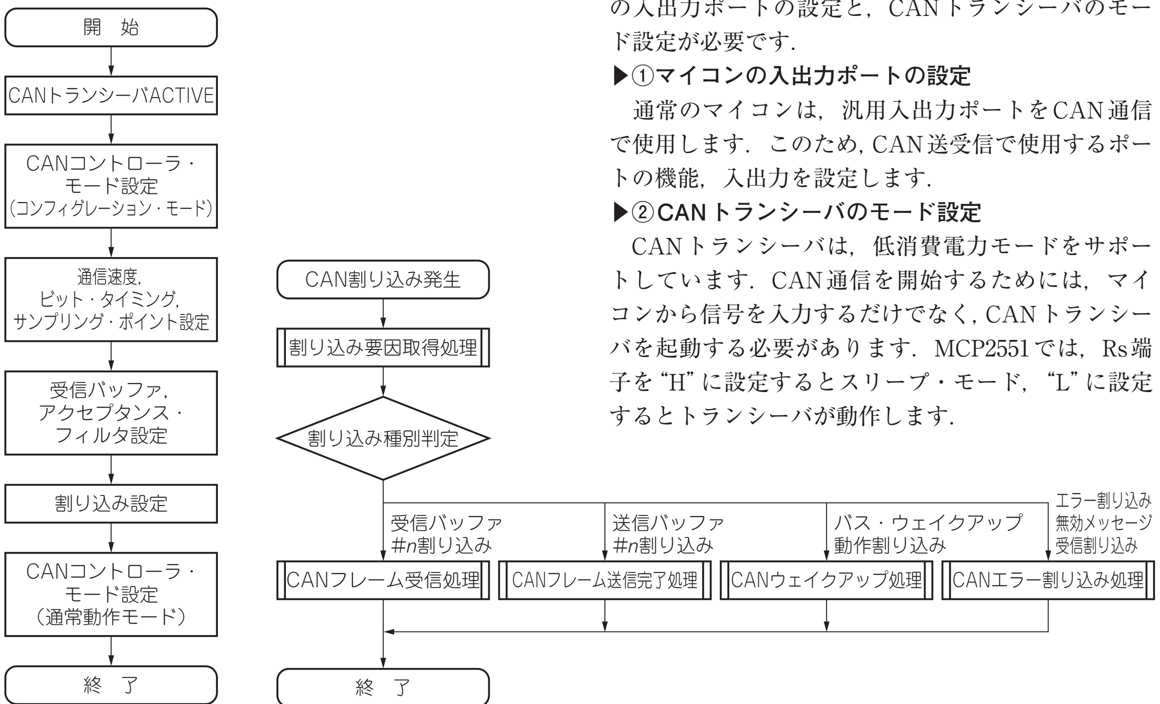


図2　CAN通信を行うための手順
今回作成するプログラムにはCANフレーム送信完了処理とCANエラー割り込み処理は含まない

表2　CANコントローラを制御するためのドライバ関数

機　能	関数名	説　明
CANコントローラ初期化関数	Can_Init()	CANコントローラのコンフィグレーションを行う
CANコントローラ・モード設定関数	Can_SetControlerMode()	CANコントローラを通信可能な状態，通信停止状態，スリープ状態(ウェイクアップ待ち)に遷移させる
CANフレーム送信関数	Can_Transmit()	CANフレームの送信を行う
CANフレーム受信関数	Can_Recieve()	CANフレームの受信を行う．割り込みハンドラから実行される関数
CANウェイクアップ関数	Can_Wakeup()	CANバス動作によるウェイクアップ時に，CANコントローラを通信可能な状態に遷移させる．割り込みハンドラから実行される関数
CAN割り込み要因取得関数	Can_GetInterruptFactor()	CAN関連割り込みが発生した場合に，割り込み要因を取得する．送受信割り込み・ウェイクアップ割り込みについては，割り込みフラグレジスタの該当フラグクリアも行う

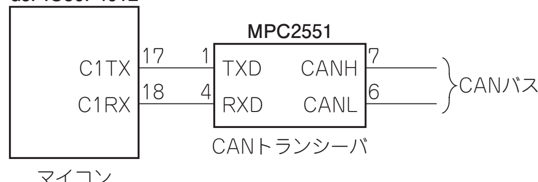


図3　マイコンとCANトランシーバとの接続

CAN通信実験回路では，Rs端子がGNDに接続されており，常時動作可能な状態となっているので，CANトランシーバのモード設定は不要です．

● ステップ2：コンフィグレーション・モードにする

CANフレームの送受信を行うためには，CANコントローラの各種設定を行い，動作状態に遷移させる必要があります．

CANコントローラは通信速度とビット・タイミングに合わせたクロックを生成するために，他の周辺機能と同様に入力クロックを選択する必要があります．入力クロックは，CAN制御とステータス・レジスタのCANマスタ・クロック選択ビット(CiCTRL.CANCKS)によって，FCYまたは4FCYを選択します．

● ステップ3：通信速度，ビット・タイミング，サンプリング・ポイントを規定に合わせる

通信速度やビット・タイミング，サンプリング・ポイントは，同一バスに接続するノード全てで合わせる必要があるため，バスに流れるフレームの設計をするときに決定されます．

前回作成したCAN通信モニタでは，自動車のOBD-Ⅱコネクタに接続してCANバスに流れるデータをモニタリングしていました．OBD-Ⅱの通信速度はISO 15765-4に定められています．今回のソフトウェアでも，この規定に準拠するように通信速度設定を行います(**表3**)．

表3　通信速度/ビット・タイミング/サンプリング・ポイントの設定

項　目	規定値
tq	100ns
fSEG1	1500ns
fSEG2	400ns
fSJW	300ns
サンプリング・ポイント	80%

(a) ISO15765-4で定められた通信速度500kbpsの設定

項　目	設定値
同期セグメント(Sync Seg)	1Tq
伝達時間セグメント(Prop Seg)	8Tq
フェーズ・バッファ・セグメント1(Phase1 Seg)	7Tq
フェーズ・バッファ・セグメント2(Phase2 Seg)	4Tq

(b) 設定値(通信速度：500kbps，20Tqの場合)

▶①入力クロックとボーレート・プリスケーラ値の設定

CANボーレート・レジスタのボーレート・プリスケーラ・ビット(CiCFG1.BRP)に設定する値を，以下の計算式で求めます．

BRP = (FCAN/(2 × Tq数×通信速度)) − 1

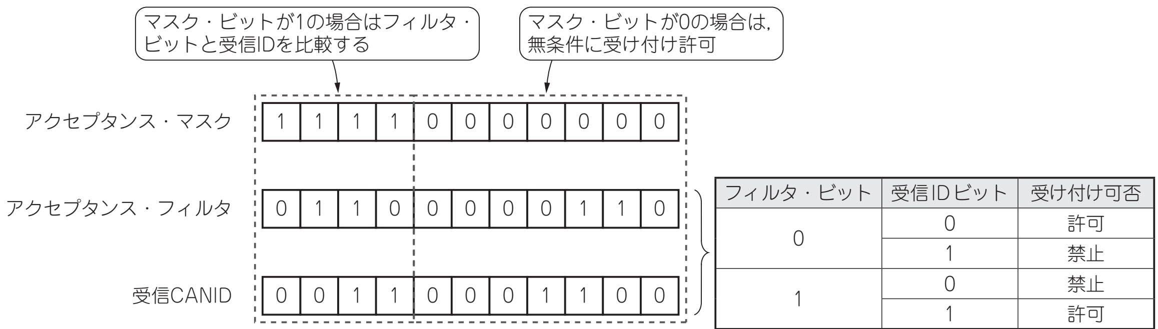
▶②ビット・タイミングの設定

Tq数，サンプリング・ポイントに合わせてボーレート構成レジスタ2の以下の値を設定します(**リスト1**)．

- 伝達時間セグメント・ビット(CiCFG2.PRSEG)
- フェーズ・バッファ・セグメント1ビット(CiCFG2.SEG1PH)
- フェーズ・バッファ・セグメント2ビット(CiCFG2.SEG2PH)

また，CANボーレート・レジスタの同期化ジャンプ幅ビット(CiCFG1.SJW)に指定の値を設定します．

同期セグメント(Sync Seg)は，1Tqを自動的にCANコントローラが確保します．



フィルタ・ビット	受信IDビット	受け付け可否
0	0	許可
	1	禁止
1	0	禁止
	1	許可

図4　ハードウェア的に受信対象のCANIDかどうかを判定するアクセプタンス・フィルタ
受信CANIDの各ビットをアクセプタンス・マスクとアクセプタンス・フィルタの設定値に従い受付可否を判断する．全てのビットが許可の場合は受信対象と見なし受信バッファに格納する

リスト1　通信速度/ビット・タイミング/サンプリング・ポイントを設定するプログラム

```
static void Can_InitControler(void)
{
    C1CTRLbits.CANCKS = 1;          /* FCAN = FCY */      ← クロック設定

    C1CFG2bits.PRSEG = 7;           //PRSEG=8Tq
    C1CFG2bits.SEG1PH = 6;          //SEG1PH=7Tq
    C1CFG2bits.SAM = 1;                                   ← 通信速度設定
    C1CFG2bits.SEG2PHTS = 1;
    C1CFG2bits.SEG2PH = 3;          //SEG2PH=4Tq

    C1CFG2bits.WAKFIL = 1;                                ← ウェイクアップ・フィルタ設定
       // Enable CAN bus Line Filter for Wake-up bit

    :
}
```

リスト2　受信バッファの選択とアクセプタンス・フィルタを設定するプログラム

```
static void Can_InitControler(void)
{
    :

    C1RX0CON = 0x0000;                                    ← 受信バッファ設定
                          // Receive Buffer1 and 0 Status
                         // and Control Register for CAN1

    C1RXF0SIDbits.SID = 0x001;
    C1RXF1SIDbits.SID = 0x008;                            ← 受信バッファ0
    C1RXM0SIDbits.SID = 0x7FF;

    C1RXF2SIDbits.SID = 0x110;
    C1RXF3SIDbits.SID = 0x220;
    C1RXF4SIDbits.SID = 0x330;                            ← 受信バッファ1
    C1RXF5SIDbits.SID = 0x440;
    C1RXM1SIDbits.SID = 0x7F0;

    :
}
```

● ステップ4：受信バッファの選択とアクセプタンス・フィルタを設定する

通常，CANバス上には複数のノードが接続し，多くのCANフレームが流れています．バス上の全てのフレームからどのフレームを受信対象とするか，CANIDで判断しなくてはなりません．

ただし，CANIDは標準IDを使用しても，2048個(0x000～0x7FF)あるため，ソフトウェアで判定する処理を実装するとかなりのCPU負荷になります．せっかく受信割り込みを使用しても，受信対象を絞り込むのに時間がかかってはもったいないですね．

そこでほとんどのCANコントローラには，アクセプタンス・フィルタという機能が備わっています．一般には，プロトコル・チェックを終えて受信バッファに格納されたフレームのCANIDに対して，ビット単位に受信対象とするかどうかを設定できます(**図4**)．

なお，dsPIC30F4012では，アクセプタンス・フィルタの設定は，受信したフレームが格納される受信バッファを選択する必要があります．

通信対象のノードが一つだけの場合は，全てのフレームを受信できるような設定でも構いません．今回は通信対象は1ノードですが，練習を兼ねてアクセプタンス・フィルタで受信フレームの絞り込みを行います(**リスト2**)．

● ステップ5：割り込みの有効/無効を設定する

dsPIC30F4012では，8個の割り込みを使用できます．

- 無効メッセージ受信割り込み
- バス・ウェイクアップ動作割り込み
- エラー割り込み
- 送信バッファ2割り込み
- 送信バッファ1割り込み
- 送信バッファ0割り込み
- 受信バッファ1割り込み
- 受信バッファ0割り込み

各割り込みの有効・無効はCAN割り込みレジスタ(CiINTE)の対応ビットによって設定します．

CAN関連の割り込みは，CAN1(結合)割り込みに集約されています．割り込みハンドラ・ルーチン内で，割り込みフラグ・レジスタ(CiINTF)，または，CAN制御とステータス・レジスタの割り込みフラグ・コード・ビット(CiCTRL.ICODE)によって割り込み要因

リスト3　割り込みを設定するプログラム

```
static void Can_InitControler(void)
{
    :
    C1INTF = 0;         // Reset all The CAN Interrupts
    IFS1bits.C1IF = 0;
            // Reset the Interrupt Flag status register

    C1INTE = 0x001F;
                  // Enable all CAN interrupt sources
                  //  IVRIE : Disabled
                  //  WAKIE : Disabled
                  //  ERRIE : Disabled
                  //  TX2IE : Disabled
                  //  TX1IE : Disabled
                  //  TX0IE : Disabled
                  //  RX1IE : Enabled
                  //  RX0IE : Enabled

    IEC1bits.C1IE = 1;   // Enable the CAN1 Interrupt
}
```

割り込みコントローラ
割り込みフラグ・レジスタ，CAN1該当フラグ・クリア

割り込みフラグ・レジスタ・クリア

各CAN割り込みの有効無効設定

割り込みコントローラ
割り込み許可制御レジスタCAN1該当フラグ・セット

を判断する必要があります．

割り込みを使ったプログラムは，メインの処理をいったん停止することになるので，共通リソースの排他処理などいろいろなことを考えてプログラムを作成する必要があります．割り込みフラグをメイン処理で定期的にチェックすることで，割り込みの対象となる事象が発生したかどうかを知ることもできますので，割り込みを使用するかどうかは十分に検討が必要です．

▶①バス・ウェイクアップ動作割り込み

通常，バス・ウェイクアップは，マイコンの各種機能を停止して，システムを低消費電力モードにするために使用します．できるだけ消費電力を抑えるために，メイン処理でチェックを行うよりも，マイコンのハード的な機能を利用してバス・ウェイクアップの検出を行う方が効果的です．

▶②エラー割り込み，無効メッセージ受信割り込み

CANプロトコル上の各種エラーを通知してくれます．エラーが発生した直後になんらかの安全処理を実施したい場合は，割り込みを使用します．

▶③送信バッファ2/1/0割り込み

送信完了したことを割り込みで通知してくれます．定期的に送信するフレームの送信間隔に精度を求める場合や，送信データ量が多く連続的に送信する必要がある場合は，割り込みを使用することをお勧めします．

▶④受信バッファ1/0割り込み

CANフレームを受信し，CANコントローラから受信データを取り出してもよいタイミングを通知してくれます．メイン処理でもチェックは可能ですが，通信速度とチェック間隔の関係を考慮することを忘れないようにしてください．通信速度よりチェック間隔が大きくなると，受信フレームの取りこぼしが発生してしまいます．もちろん，割り込み処理時間も考慮しておかないと，連続で受信した場合などは取りこぼしとなってしまうので注意してください．

今回作成するプログラムでは，バス・ウェイクアップ動作割り込み，受信バッファ1割り込み，受信バッファ0割り込みを使用します（**リスト3**）．

● ステップ6：通常動作モードにする

CANコントローラを通信可能な状態にするためには，モードを変更する必要があります．dsPIC30F4012のCANコントローラのモードを以下に示します（**図5**）．

- 通常動作モード
- 無効モード
- ループバック・モード
- リスン・オンリ・モード
- コンフィグレーション・モード
- リスン・オール・メッセージ・モード

電源投入時（リセット時）はコンフィグレーションモードで起動します．

今回作成するプログラムでは，コンフィグレーション・モード，無効モード，通常動作モードを使用します（**リスト4**）．

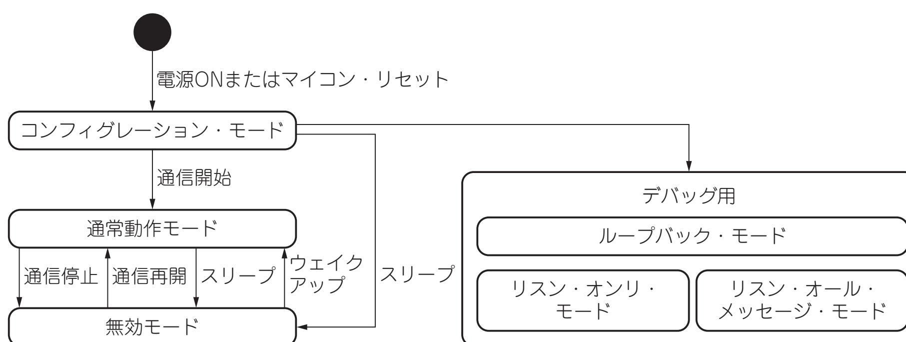


図5　CANコントローラのモード

リスト4　CANコントローラのモードを設定するプログラム

```
void Can_SetControlerMode(unsigned short mode)
{
    switch(mode)
    {
        case CAN_START:
            // Change to Normal Operation mode.
            C1CTRLbits.REQOP = CAN_OPMODE_NORMAL;
            while(C1CTRLbits.OPMODE !=
                                CAN_OPMODE_NORMAL);
            break;

        case CAN_STOP:
            // Change to Disable mode.
            C1CTRLbits.REQOP = CAN_OPMODE_DISABLE;
            while(C1CTRLbits.OPMODE !=
                               CAN_OPMODE_DISABLE);
            break;

        case CAN_SLEEP:
// Enabled Bus Wake Up Activity Interrupt Enable bit.
            C1INTEbits.WAKIE = 1;

            // Change to Disable mode
            C1CTRLbits.REQOP = CAN_OPMODE_DISABLE;
            while(C1CTRLbits.OPMODE !=
                               CAN_OPMODE_DISABLE);
            break;

        default:
            break;
    }
}
```

通信開始

通信停止

スリープ

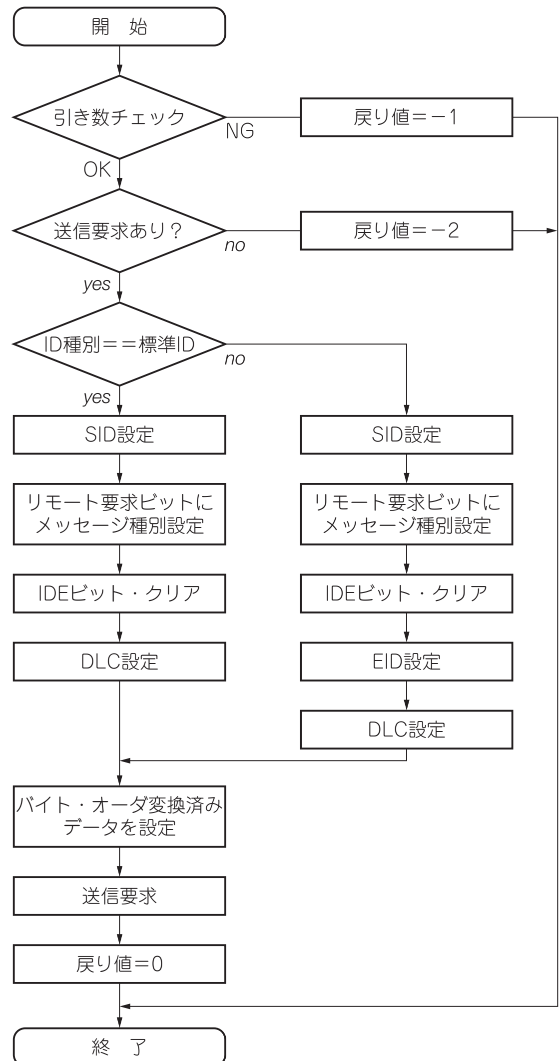


図6　送信手順

通信制御プログラム

● 送信…メッセージを書き込んで送信要求を発行

CANフレームを送信するためには，送信バッファに送信メッセージを格納し，送信要求を発行する必要があります（図6）．

送信バッファは，CANコントローラが動作モードの状態で，任意のタイミングで設定できます．dsPIC30F4012では，三つの送信バッファを保有しているため，各送信フレームの用途や発生頻度などによって，各バッファを使い分けてください．

今回のプログラムでは，送信バッファ0（TXB0）を使用した送信関数を作成します（リスト5）．

● 受信…コントローラが自動的に受信して割り込みを発生

CANフレームの受信は，アクセプタンス・フィルタ設定と受信バッファ設定のルールのもとにCANコントローラが自動的に行います（図7）．

CANコントローラが受信したCANフレームは，受信バッファに格納され，受信バッファ・レジスタの受信フル・ステータス・ビット（CiRX0CON.RXFUL）や受信割り込みによってマイコンから通知されます．

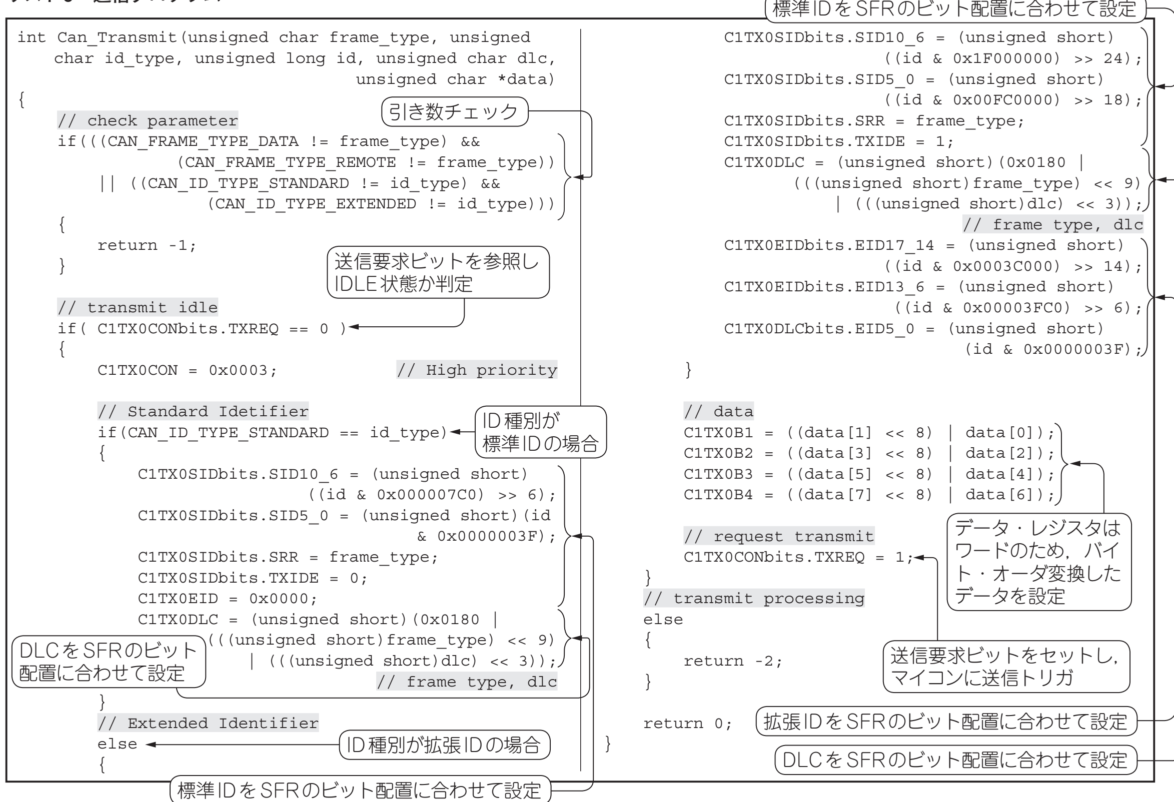
受信割り込みを使用せずに，定期的にステータスを監視する場合は，送信される頻度によって，オーバラン・エラーが発生し，受信するはずだったフレームを取りこぼす可能性がります．CANバス上に接続する機器がどのようなタイミングで送信を行うかを検討した上で，受信割り込みを使用するかどうか決定してください．

今回のプログラムでは，受信割り込みを使用した受信関数を作成します（リスト6）．

● エラー処理…コントローラが自動検出して割り込みを発生

CANコントローラは，CANプロトコルで規定された各種エラーの検出を行います．検出可能なエラーは6種類です．

リスト5　送信プログラム



```
int Can_Transmit(unsigned char frame_type, unsigned
    char id_type, unsigned long id, unsigned char dlc,
                              unsigned char *data)
{
    // check parameter
    if(((CAN_FRAME_TYPE_DATA != frame_type) &&
            (CAN_FRAME_TYPE_REMOTE != frame_type))
        || ((CAN_ID_TYPE_STANDARD != id_type) &&
              (CAN_ID_TYPE_EXTENDED != id_type)))
    {
        return -1;
    }

    // transmit idle
    if( C1TX0CONbits.TXREQ == 0 )
    {
        C1TX0CON = 0x0003;              // High priority

        // Standard Idetifier
        if(CAN_ID_TYPE_STANDARD == id_type)
        {
            C1TX0SIDbits.SID10_6 = (unsigned short)
                          ((id & 0x000007C0) >> 6);
            C1TX0SIDbits.SID5_0 = (unsigned short)(id
                                      & 0x0000003F);
            C1TX0SIDbits.SRR = frame_type;
            C1TX0SIDbits.TXIDE = 0;
            C1TX0EID = 0x0000;
            C1TX0DLC = (unsigned short)(0x0180 |
                (((unsigned short)frame_type) << 9)
                    | (((unsigned short)dlc) << 3));
                                // frame type, dlc
        }
        // Extended Identifier
        else
        {
            C1TX0SIDbits.SID10_6 = (unsigned short)
                          ((id & 0x1F000000) >> 24);
            C1TX0SIDbits.SID5_0 = (unsigned short)
                          ((id & 0x00FC0000) >> 18);
            C1TX0SIDbits.SRR = frame_type;
            C1TX0SIDbits.TXIDE = 1;
            C1TX0DLC = (unsigned short)(0x0180 |
                (((unsigned short)frame_type) << 9)
                    | (((unsigned short)dlc) << 3));
                                // frame type, dlc
            C1TX0EIDbits.EID17_14 = (unsigned short)
                          ((id & 0x0003C000) >> 14);
            C1TX0EIDbits.EID13_6 = (unsigned short)
                          ((id & 0x00003FC0) >> 6);
            C1TX0DLCbits.EID5_0 = (unsigned short)
                              (id & 0x0000003F);
        }

        // data
        C1TX0B1 = ((data[1] << 8) | data[0]);
        C1TX0B2 = ((data[3] << 8) | data[2]);
        C1TX0B3 = ((data[5] << 8) | data[4]);
        C1TX0B4 = ((data[7] << 8) | data[6]);

        // request transmit
        C1TX0CONbits.TXREQ = 1;
    }
    // transmit processing
    else
    {
        return -2;
    }

    return 0;
}
```


[受信エラー]
- 巡回型冗長チェック(CRC)エラー
- ビット・スタッフィング・エラー
- 無効メッセージ受信エラー

[送信エラー]
- アクノリッジ・エラー
- 形式エラー
- ビット・エラー
- バスオフ

発生したエラーによって，CANコントローラがどのような状態なのか，割り込みフラグ･レジスタ(C1INTF)の関連ビットを参照することによって特定できます．

エラーが検出された場合は，エラー・フレームの送信や再送，バスオフによる送信の一時的な停止など，ある程度のリカバリ処理を行う必要があります．

エラー時の処理によっては通信が途絶してしまうため，システム全体で決めていく必要があります．

▶受信時

受信時のエラーは，CANバスにノイズが発生した場合や，送信元ノードになんらかの異常が発生した場合に発生します．受信エラー発生時は，CANコントローラ上で受信したフレームは破棄されるため，ソフトウェア上では受信が途絶したように見えます．アプリケーション上では，相手からの受信が途絶した場合に代わりとなるデータで制御するなどの対処を行います．

▶送信エラー

送信時のエラーは，自分自身が原因で発生します．送信エラーが発生しているにもかかわらず送信を続けていくことは，バスを乱すことになります．一つの送信フレームに対してのエラーだけでは大きな影響はありませんが，アクノリッジ・エラーやバスオフが発生した場合は，いったん，送信を停止してください．CANコントローラを初期化することをお勧めします．

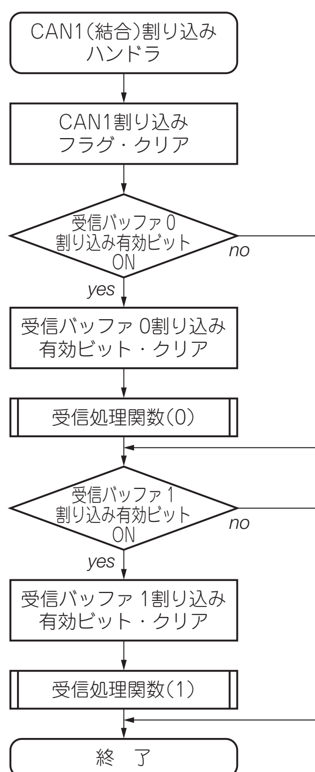
エラー・ハンドリングは複雑なものになるので，今回作成するプログラムでは，エラー状態は参照のみで，エラー発生時のソフトウェア処理は実装しません．

動かしてみる

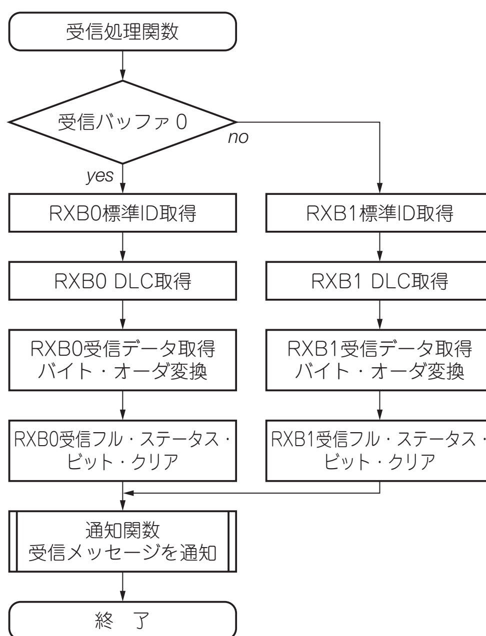
作成したCANコントローラを制御する各関数を使用して，送受信を行います(**リスト7**)．

①割り込みハンドラ・ルーチンの実装

CAN1(結合)割り込みハンドラ・ルーチンから，CAN割り込み要因取得関数を実行します．戻り値で取得した要因に従って，CANフレーム受信関数，



（a）割り込みハンドラ



（b）受信処理

図7　受信手順

CANウェイクアップ関数を実行します．

②CANコントローラの起動

CANコントローラ初期化関数を実行し，アプリケーションが通信開始したいタイミングで，CANコントローラのモード設定関数を実行します．

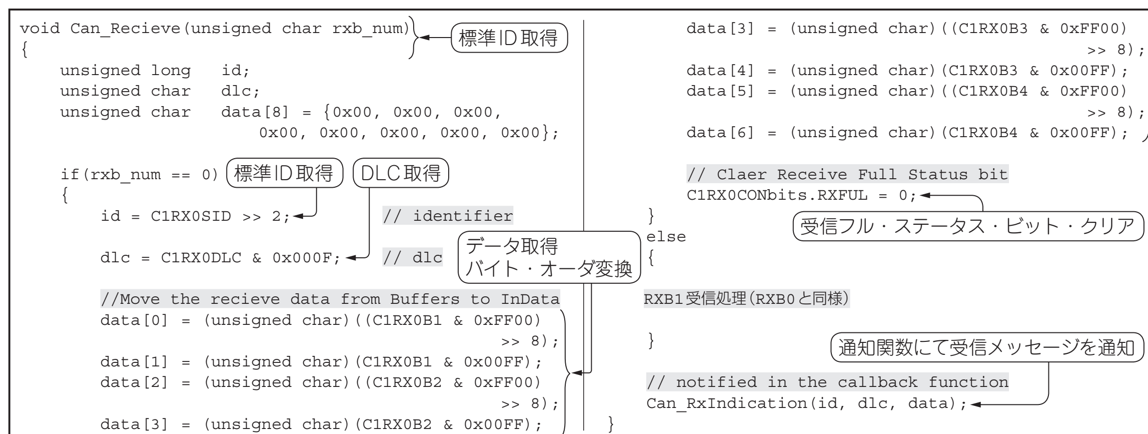
③フレームの受信

割り込みハンドラ・ルーチンで実行するCANフレーム受信関数から，CANフレーム受信通知関数が実行されます．CANフレーム受信通知関数は，ドライバ・ソフトウェアからのコールバック関数となるため，アプリケーション側で関数実体の実装を行います．受信したフレームが引き数として通知されるため，受信フレームをアプリケーション側にコピーします．今回のサンプルでは，受信したフレームに対して，CANIDをインクリメントし，データを反転したフレームをオウム返しに送信します．

④フレームの送信

アプリケーションで送信したいタイミングで，CAN

リスト6　受信プログラム



```
void Can_Recieve(unsigned char rxb_num)
{
    unsigned long   id;
    unsigned char   dlc;
    unsigned char   data[8] = {0x00, 0x00, 0x00,
                       0x00, 0x00, 0x00, 0x00, 0x00};

    if(rxb_num == 0)
    {
        id = C1RX0SID >> 2;             // identifier

        dlc = C1RX0DLC & 0x000F;        // dlc

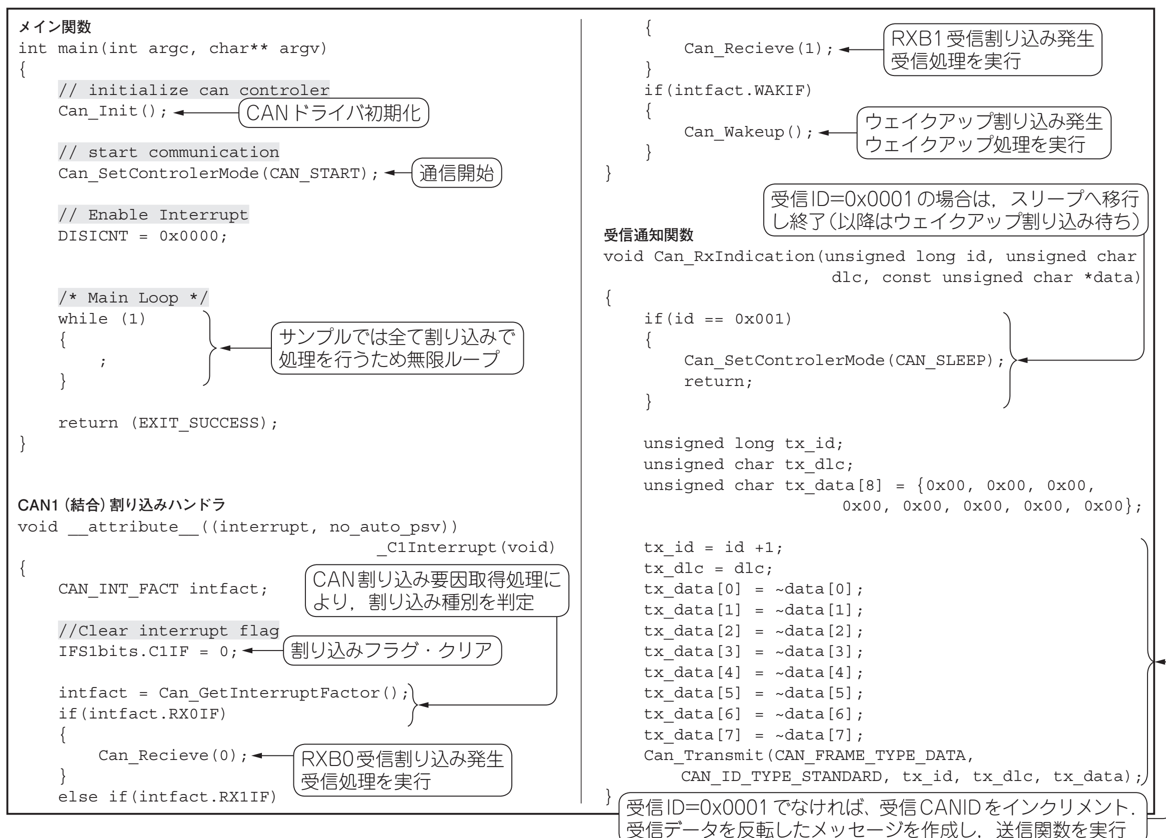
        //Move the recieve data from Buffers to InData
        data[0] = (unsigned char)((C1RX0B1 & 0xFF00)
                                    >> 8);
        data[1] = (unsigned char)(C1RX0B1 & 0x00FF);
        data[2] = (unsigned char)((C1RX0B2 & 0xFF00)
                                    >> 8);
        data[3] = (unsigned char)(C1RX0B2 & 0x00FF);
        data[3] = (unsigned char)((C1RX0B3 & 0xFF00)
                                    >> 8);
        data[4] = (unsigned char)(C1RX0B3 & 0x00FF);
        data[5] = (unsigned char)((C1RX0B4 & 0xFF00)
                                    >> 8);
        data[6] = (unsigned char)(C1RX0B4 & 0x00FF);

        // Claer Receive Full Status bit
        C1RX0CONbits.RXFUL = 0;
    }
    else
    {
    RXB1受信処理(RXB0と同様)

    }

    // notified in the callback function
    Can_RxIndication(id, dlc, data);
}
```

リスト7　送受信を行うメイン・プログラム



```
メイン関数
int main(int argc, char** argv)
{
    // initialize can controler
    Can_Init();

    // start communication
    Can_SetControlerMode(CAN_START);

    // Enable Interrupt
    DISICNT = 0x0000;


    /* Main Loop */
    while (1)
    {
        ;
    }

    return (EXIT_SUCCESS);
}


CAN1（結合）割り込みハンドラ
void __attribute__((interrupt, no_auto_psv))
                                    _C1Interrupt(void)
{
    CAN_INT_FACT intfact;

    //Clear interrupt flag
    IFS1bits.C1IF = 0;

    intfact = Can_GetInterruptFactor();
    if(intfact.RX0IF)
    {
        Can_Recieve(0);
    }
    else if(intfact.RX1IF)
    {
        Can_Recieve(1);
    }
    if(intfact.WAKIF)
    {
        Can_Wakeup();
    }
}


受信通知関数
void Can_RxIndication(unsigned long id, unsigned char
                    dlc, const unsigned char *data)
{
    if(id == 0x001)
    {
        Can_SetControlerMode(CAN_SLEEP);
        return;
    }

    unsigned long tx_id;
    unsigned char tx_dlc;
    unsigned char tx_data[8] = {0x00, 0x00, 0x00,
                        0x00, 0x00, 0x00, 0x00, 0x00};

    tx_id = id +1;
    tx_dlc = dlc;
    tx_data[0] = ~data[0];
    tx_data[1] = ~data[1];
    tx_data[2] = ~data[2];
    tx_data[3] = ~data[3];
    tx_data[4] = ~data[4];
    tx_data[5] = ~data[5];
    tx_data[6] = ~data[6];
    tx_data[7] = ~data[7];
    Can_Transmit(CAN_FRAME_TYPE_DATA,
        CAN_ID_TYPE_STANDARD, tx_id, tx_dlc, tx_data);
}
```


フレーム送信関数を使用して送信を行います．フレーム種別（リモート・フレーム，データ・フレーム），ID種別（標準ID，拡張ID）を設定することが可能です．今回のサンプルでは，データ・フレームと標準IDの設定で，1000ms周期での周期送信とCANフレーム受信通知関数内での応答送信を行います．

⑤スリープ・ウェイクアップ

スリープ・タイミングで，CANコントローラ・モード設定関数を実行します．ウェイクアップ時の通信再開処理は，CAN1（結合）割り込みハンドラ・ルーチンから実行する，CANウェイクアップ関数内で行われます．

＊　＊　＊

これまで9回にわたって，CANの紹介から簡単な送受信プロクラムの作成まで紹介してきました．

CANは工夫によってはとても便利な通信規格です．また，CANは自動車業界では信頼性の高い有線ネットワークとして定着しており，標準ソフトウェア規格も公開されています．堅牢な通信システムを構築するのであれば，これらの規格に準拠した通信ソフトウェアを利用することで高品質なネットワークを構築することができます．

組み込み機器間での通信を行う場合には，ぜひ，CANを選択肢の一つとしてみてください．

◆参考・引用*文献◆

(1) MPLAB X 統合開発環境（IDE），Microchip Technology.
http://www.microchip.com/ja/mplab/mplab-x-ide
(2) MPLAB XCコンパイラ，Microchip Technology.
http://www.microchip.com/ja/mplab/compilers
(3) MPLAB IPE（統合プログラミング環境）ユーザガイド，Microchip Technology.
http://www.microchip.co.jp/download/dl_download.php/ID=08732df466905960761786 4626e738e7091e4fdf/
(4) PICkit 3 In-Circuit Debugger，Microchip Technology.
http://www.microchip.com/DevelopmentTools/ProductDetails.aspx?PartNO=pg164130

たかしま・ひかる

生業プログラマ必読! 安心を作る考え方&しくみ

宇宙船の設計で考える?!

研究! モノづくりの最新コモンセンス「機能安全」

最終回 第12回

最初に全部決めるのが最重要…機能安全マネジメントFSM

森本 賢一

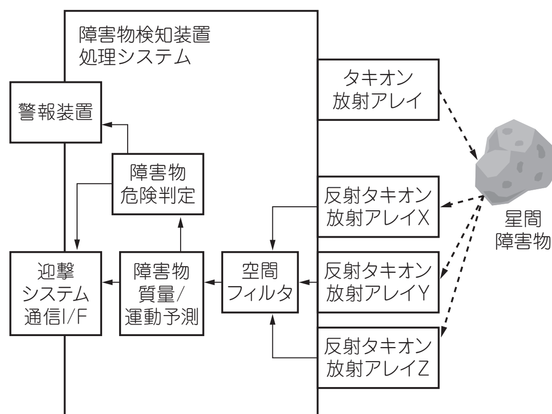


図1　今回のターゲット装置…障害物検知装置(以前設計)



図2　ほんのちょっとの式の違いでもアルゴリズムのミスは危な過ぎる…システマティック故障

● 今回のテーマ…ソフトウェアによるシステマティック故障を低減するための基本

いざというときに確実に動作する，信頼性(安全完全性レベル：SIL)の高いシステムを開発するためには，ハードウェアとソフトウェアを俯瞰したシステムの構想設計がとても重要です．

システムにはランダム・ハードウェア故障と，システマティック故障の2種類があります(これまでの連載でも紹介)．これまで解説してきたのは，主にランダム・ハードウェア故障についての対策でした．

今回最終回は，機能安全規格において，システマティック故障をどのように低減し，安全完全性を高めようとしているのかを解説していきます．

ほんのささいなことで痛恨の一撃…なんて恐ろしいシステマティック故障

● 今回のターゲット装置…障害物検知装置

図1は今回のターゲットである障害物検知装置の機能ブロック図です．タキオンの反射スペクトルの分析によって，障害物の質量や運動量，および相対的な速度を算出し，警告・回避・迎撃のいずれかの判断を下します．以前の連載では，タキオン検知アレイや放射アレイに故障が発生したケースへの対処を解説しました．今回はシステマティック故障としてどのような故障があり得るか考えてみましょう．

● あり得るシステマティック故障の例…原理・アルゴリズムの間違い

障害物の質量や運動方向を予測する物理モデルは，さまざまな法則に基づき設計されています．計算は極めて高度な数学を使用しますから，ほんのささいな計算アルゴリズムの間違いが，大きな誤差となって間違った予測を導く可能性があります(**図2**)．速度が速くなるほど誤差が大きくなるような間違いならば，初期の試験航行では問題が発見できないかもしれません．

クリティカルなシチュエーションは試験として実施できない場合もあります．例えば無重力かつ光速移動の場合に初めて露呈する間違いは，地上の試験で事前に確認することは困難です．

機能安全マネジメントFSM…システマティック故障を低減する

● 機能安全の最も重要な設計思想…最初に活動から人材まで全部決める

原理・アルゴリズムでの間違いを防止するために

は，その物理法則やアルゴリズムに詳しい人物を多数そろえて，解釈や近似などの設計プロセスを評価するべきでしょう．機能安全の規格では，SILが高いシステムは，評価体制も要求が高くなります．

審査(Assessment)は，例えばSIL1では最低限異なる人物，SIL2では異なるセクション・組織などの要件です．それら対になる評価者が，設計者と同じレベルの技術者であることを求めています．Assessmentの他にも，検証(Verification)/妥当性評価(Validation)という活動も求めていますが，これも設計者と独立して行うことを求めています．

このような設計や評価の取り組み方を，機能安全ではプロセスと呼び，安全システムの立案，開発，運用全てのプロセスについてルールを取り決めることを求めています．そのルールのひとまとめをFSM(Functional Safety Management)といい，それを図書にしたものをFSM計画書(FSM PlanまたはSafety Plan)といいます(**図3**)．機能安全ではあらゆる活動に先立って，このFSM計画書をまとめることが必要です．FSM計画書では，関係する全ての活動を定義し，その責任者を決め，活動と活動の関係やその成果(図書)について規定します．文字通り全ての活動や資源(設備や人材)がプロセスとしてつながっていることを最初に示すイメージです．

● 重要なこと…途中経過も漏れなく管理するトレーサビリティ＆変更管理

FSMの中で特に注意すべきは，構成管理とトレーサビリティ，そして変更管理です．機能安全では図書を何度も何人もの人でチェックし改訂することになります．おのずと多くの図書がそれぞれバラバラなバージョンとなることでしょう．その場合，最終的に完成したシステムを表す図書は，どのようなバージョンの図書の組み合わせなのか分からなくなる可能性があります．

このようなことを防ぐために，全体のバージョンの構成を常に観測できる必要があります．ハードウェア基板の版数やソフトウェア・ライブラリのバージョン構成も同様です．試験を実施したら，実施した特定の構成に対しての結果を保管します．構成が異なるものには，その試験結果は無効ですよね．当然新しい構成に対して再度，試験をしなければなりません．

図書が「製造図」の場合，多くは「最新の図書が常に正しい」というのが現場のイメージではないでしょうか．工場での製造のための図面の場合，しかしまれに複数の関係する図書で，互いに矛盾する変更がなされる場合があり得ます．また他方がまだ古いままということもあるでしょう．このようなささいな間違いや食い違いの中に，システマティック故障が紛れ込む可能性があると考えるべきです．

そこで，機能安全では構成管理に加え，図書と図書の間の記述の一致を厳密にチェックすることを求めています．これをトレーサビリティといいます．その上で一貫性を示す必要があります．構想設計で取り決めた戦略を，実装や試験段階の評価基準まで一貫しているのかを確認するために，試験の判定基準から実装段階，構想設計段階まで，記述をたどれることを求めるわけです(**図4**)．

もしも途中で間違いに気づいた場合，または出荷後に変更が発生した場合に，すぐに該当箇所を修正してはなりません．これを変更管理といいます．変更が必要な事態になった場合，まずは変更するべき図書や個所を明確にした後，あらかじめ決めたメンバで，変更影響分析(Impact Analysis)を実施します．該当箇所の修正が他にどのような影響を与えるのか，関連して修正するべき設計情報があるのか，再試験が必要なのはどの項目かなどを全て洗い出し，レポートを作成します．その結果を関係者で共有し，責任者が関係者に変更作業の開始を宣言します．

実際の変更はこの宣言の後にしなければなりません．「この変更はすごくささいだから」と，担当者が

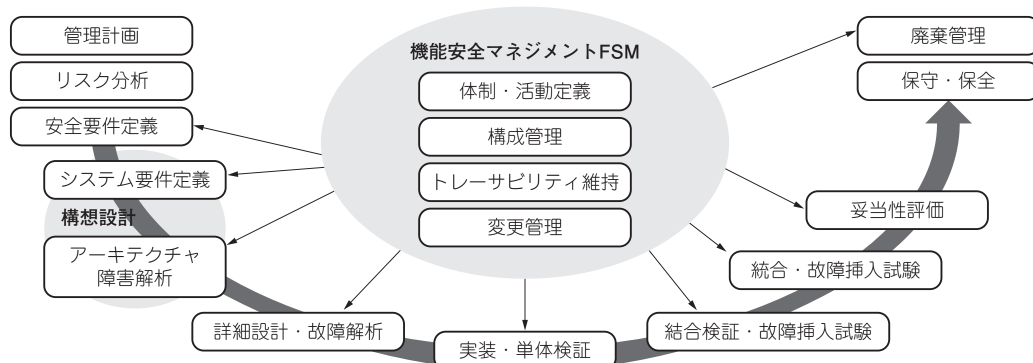


図3　全ての活動を最初に規定する…機能安全マネジメント(FSM)

第6回　購入部品のリスク評価＆対策(2016年1月号)
第7回　システムの安全性を診断する方法(2016年4月号)
第8回　システムの安全性を評価する方法②…危険側故障確率*PFD*(2016年5月号)

自分だけの判断でこっそり修正するのはご法度です．このような変更が発生したときの処理の流れや，責任者，用いる帳票などに対しても，FSMであらかじめ取り決めます．

● 途中経過まで全部管理するのはいやがらせってわけじゃない…背景にある製造物責任（PL）

構成管理，トレーサビリティ，一貫性にしろ，変更管理にしろ，成果物ではなく途中段階の経過情報に過ぎないと感じるかもしれません．最終的に「品質の高いものが完成すればよいのではないか」という観点に立てば，とても無駄な作業に感じるかもしれません．

ましてや開発作業の前に，これからのプロセス全体を計画書として用意してから取り組むなんて，すごく効率の悪い開発に思えるでしょう．しかし，現実の欧米において安全に関わる仕組みは，このような進め方が一般的です．彼ら自身が普段行っているスタイルです．

機能安全規格の背景には，製造物責任（PL：Product Liability）があります．PLには法的な要求事項（Legal Requirement）が守られているのかという観点に加え，最大限に安全に配慮したのかという観点（State of the art）が問われます．

安全関連で求められる書類に，Safety Caseというものがありますが，Safety Caseとは極端にいえば製造物責任を裁判で問われた際に，この二つの観点がどのように扱われて設計されたのか，その上でどのように製造・検証されたのかを証明する図書のパッケージです（**図5**）．

Safety Caseは，完成した物品そのものではありませんし，製造のための図面集でもありません．ましてや，その莫大な製造図面を理解するための後付けの解説書でもありません．

● 最初に全部決めないといけない理由…後からじゃ途中経過を証明できない

大まかにいえば，前述のFSM計画書はSafety Caseの目録になります．FSMで規定した活動の成果物が最後にまとまってSafety Caseと呼ばれるものとなります．この連載で多くのページを割いたリスク分析やシステムの障害解析の記録は，Safety Caseの一部です．多様な人で構成されるチームで，ルールを決めて網羅的に障害を洗い出した経緯，記録，努力こそが，State of the artの証明となるものです．

従って，開発完了後にSafety Caseを作成することはほぼ不可能です．どのような体制や資源（設備や人）で，どのような活動をして，どのような記録を残すのか，取り組みの戦略をあらかじめ取り決めておかないと，Safety Caseは作りようがありません．このような背景から，IEC 61508に限らずさまざまな機能安全



図4　活動がちゃんと全部たどれることが重要

規格では，まずFSMを規定することを求めるのです．

FSM計画書を事前にまとめず，さらに製品ができてからSafe Caseを作成する行為は，規格へのコンプライアンス以前に，背景にある欧米の技術者の文化からは，「夏休みの宿題を始業式前日に無理くり終わらせた」という感覚です．内容の妥当性はまず評価されません．

FSMのソフトウェア開発規定

FSMは全プロセスを対象としますから，ハードウェアの設計ルールも，ソフトウェアの設計ルールもどちらも規定しなければなりません．そのうち特に重要なのは，ソフトウェア開発ルールの規定です．

● ソフトウェアは最初にプログラム以外の方法で表現する

バグのないソフトウェアを作る，これは永遠の課題です．機能安全IEC 61508では，ソフトウェアの開発



図5　証明図書（Safety Case）は恥ずかしくても全部見せないといけない

プロセスにさまざまな工夫や努力をするように求めています．大まかにいえば，要件定義，アーキテクチャ設計，コンポーネント設計，モジュール設計と階層的に詳細な設計を進め，その設計階層ごとに対応した試験を実施してくみ上げていくプロセス，V字型の開発プロセスの徹底です．その階層ごとに適切な図書をしっかり書いて，最終的なC言語やアセンブラのプログラム・コードの作成に進めるイメージです．いきなりプログラムを書いてしまうのはご法度です．

プログラム言語のルールを知っていれば，作者以外でもある程度は読むことはできます．しかし莫大なプログラムの中に潜む，作者すら気づかない矛盾や設定の不整合を読み取ることは難しいでしょう．このため，Vモデルの各階層で，作ろうとするプログラムの構造を「プログラム以外の手段で」表現し，作者以外の人でもチェックしやすくします．もちろん，作者自身も自分で気づく可能性が高まります．このようなことから，機能安全では完全性の高いプログラム設計の設計図書は，形式手法や汎形式手法での作成を求めています．

● 最初に設計ルールや開発体制を決める

ソフトウェアの設計図書を作成するためのルールを，デザイン・ルール/コーディング・ルールといいます．機能安全におけるソフトウェア開発では，まずはこのルールの確立，担うチームの教育や管理などのマネジメント体系について明確に取り決めることから始めます．まさに先ほど解説したFSM計画書と同様なことを事前に取り決めてから取り組むこととなります．ソフトウェアに潜在する故障は，まさにシステマティック故障ですから，その低減はプロセスを厳密に規定し，しっかり守ること以外にありません．プログラムの完全性は，プログラムを書く前が大切です．

このようなソフトウェアの開発プロセスやマネジメント体系は，企業ですでに取り組まれているところも多いでしょう．例えばCMMIのレベルに応じたチーム能力の評価や教育の基準があれば，それに基づけばよいです．機能安全向けに新たに社内ルールを作り直す必要はありません．もちろん，完全性レベルに応じてIEC 61508で求められることが変化しますから，不足する要件は自分たちのルールに追加することで対応します．例えばコーディング・ルールが社内ルールしかないケースでは，MISRAなどの一般的な組み込みコーディング・ルールを追加します．作成したコードを自動ツールでチェックするなどの仕組みの追加が必要かもしれません．

● ルールを決めて守ることが大切

大切なことは，FSM同様に開発作業を始める前に，必要な活動，その責任者，活動と活動の連携，デザイン・ルール/コーディング・ルールや教育などのマネジメント体系を取り決めることです．もちろん，その中には構成管理・トレーサビリティ，変更管理を含めなければなりません．プログラムこそ，担当者が勝手に修正を加えがちです．評価試験の最中に顕在化したバグを，「単純なバグだから」とささっと修正して試験を継続するなどもってのほかです．図書と同じく，変更影響分析（Impact Analysis）を実施し，ルールに基づいて修正作業に入りましょう．その変更の履歴も，最終的なSafety Caseを構成する一部となります．

＊　＊　＊

機能安全が目指すものは機械や設備のリスク低減です．そのための仕掛けの多くが，組み込みコンピュータが担う現代，設備のリスクから安全システムのソフトウェア，電子部品の一つにまで，安全を実現する一貫した戦略と，その戦略を実現するモノづくりの仕方が必要なことを解説してきました．FSMや構想設計（アーキテクチャ・システム障害解析）を重視する機能安全の背景についてもお話ししました．あまりなじみのない構想設計にこそ，機能安全を攻略する重要なポイントがあるとご理解いただければ嬉しいです．

一部の電気製品事業から撤退する日本企業を指して，電気電子技術は日本の技術者が深める分野ではない，海外から購入すればよいとの論調もあるようです．それは大きな誤解です．自動車の自動運転やドローン，ロボットなど，まさに日本のモノづくりの幅の広さ奥深さが生きる製品や事業が広がってきています．そのいずれにも機能安全のアプローチが不可欠です．「品質世界一の日本」が，「安全性世界一の日本」でもあるために，組み込みコンピュータの信頼性技術の深化は欠かせません．

機能安全規格が示唆するのは，そのような信頼性は低減すべきリスクに基づき精密に設計されるべきものだということです．決して買ってきて組み立てるだけでは達成できないものです．この12回の連載を通じ，本誌読者の方々，組み込みコンピュータ・エンジニアの方々が担う役割の大きさを，あらためてお伝えできたならば幸甚です．これからも組み込みシステム，制御システムの信頼性・完全性の向上を共に取り組んでいきましょう．

◆参考文献◆

(1)（株）制御システム研究所のホームページ，
http://controlsystemlab.com/

もりもと・けんいち

Information

新製品のリリースをお送りください! ～掲載料金は無料です～

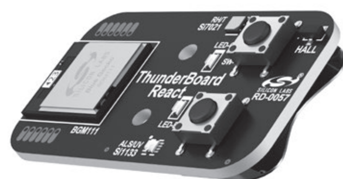
製品の特徴，価格（サンプル価格でも可），編集部および読者の問い合わせ先（電話番号，メール・アドレス）を明記の上，写真（データ可），可能ならデータシートを添付し，右記へご送付ください．

〒112-8619 東京都文京区千石4-29-14
CQ出版株式会社
インターフェース編集部 新製品担当
newsinter@cqpub.co.jp

温湿度/6軸センサ＆ARM Cortex-M4内蔵BLE無線モジュール搭載ボード　Thunderboard React

シリコン・ラボラトリーズは，無線センサ・ノード開発用のプラットフォームを発売した．センサ相対温湿度(Si7021)，UV指数/環境光(Si1133)，6軸加速度/角速度(InvenSense社MPU-6500)，磁気(Si7201)を計測できる各センサと，ARM Cortex-M4マイコン内蔵のBluetooth Smartモジュール(BGM111)，2個のスイッチ，2個のLEDを搭載する基板やソフトウェアなどが含まれる．基板の外形寸法は44mm×25mm．ボタン電池(CR2032)で動作する．基本キット(Board Kit)の他，木製4輪ミニチュア・カーが付属するCar Kitがある．

■ 価格：29ドル(Board Kit)
■ シリコン・ラボラトリーズ
TEL：(03) 5460-2411



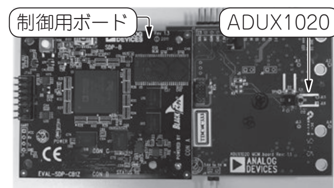
http://www.silabs.com/

わずか3.49ドルで15cmの範囲の高精度ジェスチャ認識を行える光学センサ　ADUX1020

アナログ・デバイセズは，1個のセンサで20cmまでの距離や，0.5cm～15cmの範囲のジェスチャを測定できる光学センサを発売した．外付け用の赤外LEDドライバ，14ビットA-Dコンバータ，ジェスチャ・エンジンなどを集積する．ホスト・インターフェースはI^2C．電源電圧は1.8V．パッケージは，2mm×3mmの8ピンLFCSP．

ADUX1020を搭載するボードと，Blackfinプロセッサを搭載する制御用ボードなどで構成される開発キットが用意される．

■ 価格：3.49ドル(ADUX1020BCPZ)，119ドル(開発キット)
■ アナログ・デバイセズ(株)
http://www.analog.com/jp/about-adi/contact-us.html

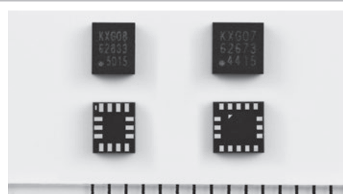


http://www.analog.com/jp/

消費電流が0.6mAと低い6軸加速度/角速度センサ　KXG07，KXG08

ロームは，2.5/3mm×3mm×0.9mmと小型で，動作時の消費電流が最大0.6mAと低い，Kionix社の6軸加速度/ジャイロ・センサを発売した．2/4/8/16gの加速度を3軸と，±2048/1024/512/256/128/64°/sの角速度を3軸，それぞれ16ビット解像度で検出できる．ホスト・インターフェースはI^2CおよびSPI．マイコンとの同期機能や4Kバイトのバッファ・メモリ，外部センサからのデータの取得機能のためのI^2Cマスタ機能，温度センサ，電圧レギュレータを搭載する．電源電圧は1.35V～3.3V．

■ サンプル価格：500円
■ ローム(株)
https://www.rohm.co.jp/web/japan/contactus



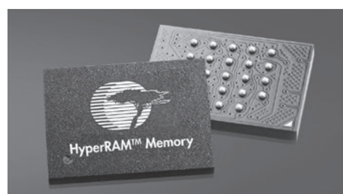
http://www.rohm.co.jp/

フラッシュ・メモリと同じ高速バスにつなげる64MビットDRAM　HyperRAM

サイプレス セミコンダクタは，12本と少ないインターフェースHyperBusに対応する64Mビットのセルフ・リフレッシュDRAMを発売した．リード/ライト性能は最大333Mバイト/s．3V動作品(S27KL0641)と1.8V動作品(S27KS0641)がある．パッケージは，6mm×8mmの24ピンBGA(Ball Grid Array)．動作温度範囲は－40～＋85/105℃．

HyperBus採用のフラッシュ・メモリとしてHyperFlashがある．一つのバス上で接続できるため，システム設計の簡素化や基板コストの削減につながる．

■ サンプル価格：下記に問い合わせ
■ 日本サイプレス(株)
TEL：(044) 920-8108

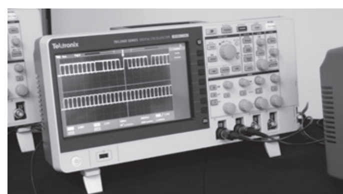


http://japan.cypress.com/

9型横長画面によって時間方向の信号変化が見やすくなった低価格オシロスコープ　TBS2000シリーズ

テクトロニクスは，トリガやFFT解析，自動計測，カーソルといった信号の観測・計測についての基本性能を強化した低価格オシロスコープを発売した．9型ディスプレイを搭載し，水平15divの表示ができる．周波数帯域は70MHzまたは100MHz．それぞれ2チャネル品と4チャネル品がある．サンプリング周波数は1GSps(2/4チャネル品の1/2チャネル使用時)で20Mポイントの長時間記録ができる．ウェブ・サーバ機能により，離れた場所からの制御が可能．外形寸法は，372.4mm×174.9mm×103.3mm．

■ 価格:148,000円(70MHz，2チャネル)～274,000円(100MHz，4チャネル)
■ テクトロニクス社
TEL：0120-441-046



http://jp.tek.com/



読者の声

＜特集＞8月号　SDカード便利帳付き！ 切手サイズIoT無線センサ入門

- SDカードがマイコンとして使えるようになるということを初めて知って驚きました．（しょうた）
- FlashAirの存在は前から知っており，画像をWi-Fi経由で送れる便利なデバイス程度にしか考えていなかったが，今号を読むとGPIO・SPIが使えたりLuaが使えたりと考えていたものとかなり違い，その多機能性に驚いた．またNFCの解説なども読むとSDカード単体で多様な利用方法があり，単なるストレージにとどまっておらず非常に面白く読めた．（匿名希望）
- 知人の技術者にAIよりはCPUの小型化がテクノロジとして脅威になるだろうと予測している人がいる．先月号以上に興味深い話題．（シンゴジラ）
- 「保存版 SDカード便利帳」では規格の違いや速度の比較などがわかりやすくまとまっており読みやすかった．特に内部レジスタやコマンド，状態遷移についての解説が非常に参考になった．（匿名希望）
- FlashAirの存在は知っていたが，あまり魅力に感じていなかった．今回の記事を読み，応用例を検討する価値があるかもと思えた．開発者向けWebサイトがあることも有益な情報でした．（アンダルシア）
- 取得したデータの外部への送信が，見た目追加機器なしの感覚で行える時代に驚いています．私は昔50baudの回線でやり取りしていました．どれだけデータを絞ったか．（匿名希望）
- 数年ぶりに購入しました．Lua言語の記述にビビッと来たのがきっかけです．言語仕様や処理系の解説がなかったのが残念でした．またの機会にお願いします．（@ヒデ）

＜Interfaceについて＞

- ESP-WR00M-02は使い勝手が良さそうですね．しかし，技適済みのWi-Fiモジュールがこれほど安価に入手できるようになるといろいろ楽しみですね．（出玉のタマ）
- とても面白いです．少々，学生である自分にでは実力不足を感じさせられることがありますが，これはこれでとても良い勉強となっています．（＾◇＾）
- 最近のマイコン評価キットは，何でもてんこ盛りですごいですね．ミドルウェアまでついてきて，動作保証付きは素晴らしいです．一昔前には，こんなに至れり尽くせりなものはなく，動かすまでに苦労することもありましたが，それはそれで楽しくもあり勉強にもなり，身に付くことがたくさんあったなぁ～と思ったりもします．（匿名希望）
- 無線センサのような，部品ともシステムとも言いにくいものは意外と取り上げられない傾向なので，今後ともニッチ？な分野の紹介をお願いします．（匿名希望）
- Interfaceは創刊時代から15年ほど毎月購入していましたが，最近は書店やWebで見て興味のあるとき購入しています．昔は，確か8008の記事があったと記憶してます．そのころのIC価格で千倍以上の能力のCPUが発売され，いろいろなことができて最高ですね．（北海の熊）

デバッグ

本誌のバック・ナンバに下記の誤りがありました．お詫びして訂正いたします．（編集部）

＜2016年9月号＞

●緊急特集 新型ラズパイ・カメラ2

- p.102　**写真2**中の吹き出し：2152画素→2512画素
- p.103　**表1**中の総画素数：3296×2152→3296×2512

●生体センシング実験室

- p.127　**図2(c)**：「After」は下側の波形でなく上側の波形．下側の波形は上側の波形と比較するため別途用意した心電図

投稿歓迎！

本誌に投稿をご希望の方は，連絡先を明記のうえ，テーマ，内容の概要をレポート用紙1～2枚にまとめてご送付ください．

送り先：〒112-8619 東京都文京区千石4-29-14
CQ出版株式会社 Interface投稿係
E-mail：supportinter@cqpub.co.jp

編集者募集！

→p.179を見て！！

組み込みもん++ 36

米田 裕 PRESENTED BY YUTAKA YONEDA



2016年8月号の記事ベスト3

＜特集＞ 切手サイズIoT 無線センサ入門

第1位　第1章　マイコンみたいに使えるWi-Fi付きSDカードFlashAir初体験

第2位　第3章　IoTにピッタリ!? 切手サイズ・コンピュータの可能性を探る

第3位　第2章　切手サイズWi-Fiモジュールで超小型センサ端末を作る

＜解説/連載記事＞

第1位　保存版 SDカード便利帳

第2位　ディープ・ニューラル・ネットワーク×FPGA入門〈第1回〉FPGAがディープ・ニューラル・ネットワークに向く理由

第3位　Max240MHz! ルネサスの本気ARM Cortex-Mマイコン誕生

お知らせ

■ お問い合わせ先のご案内

- 在庫，バックナンバ，年間購読送付先変更に関して ― 販売：03-5395-2141
- 広告に関して ― 広告：03-5395-2131
- 雑誌本文に関して ― 編集：03-5395-2122

記事内容に関するご質問は，返信用封筒を同封して編集部宛てに郵送してくださるようお願いいたします．筆者に回送してお答えいたします．

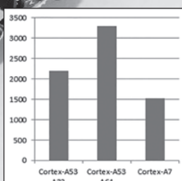
11月号予告 2016年9月24日発売

特集 64ビット並列時代 ラズパイ3ではじめる ARMコンピュータ

64ビット並列アーキテクチャの研究

Cortex -A53　Cortex -A53　Cortex -A53　Cortex -A53

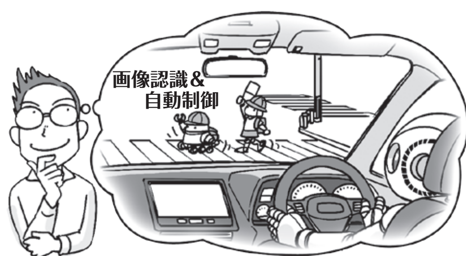
ARM直伝!



64ビットARMコアの実力

最大32枚(128コア)スーパコンピュータ





超高性能手のひらコンピュータで広がる夢

編集後記

●自分の地元と違って，東京では7月から夏祭りをやっています．町が多いので，8月に入っても毎週近所のどっかで夏祭り．秋になると収穫祭とか地域振興フェスタ的なのがあって，東京はほんま祭りばかりやってんなって感じです．(54)

●梅雨も明けて本格的な夏の到来です．去年は一番暑い時にエアコンが壊れて急遽買い替え．取り付け工事の順番待ちなどで1週間ほどリビングには人が住めませんでしたが(笑)，今年は(室内では)快適に過ごせそうです．(M)

●話題のスマホ・ゲームが配信された直後の週末，お出かけ先の公共施設で異様な光景を目にしました．週明け，もともと人が多い都内ではあまり気にならないと思っていたら，弊社内に何人ものトレーナが生まれていました．(N2)

●ポケモンGOやってみました．記事ネタ調査のためです．前回のウェアラブル・センサ特集の時にも思ったのですが，9軸センサやGPSって本当に凄いですね．日本から次世代のスマホが生まれないかな．ガンバレ日本!(仏)

●残暑が厳しく日中は体温より気温が高い日が続いている．暑さに負けない体力作りも必要と感じているが，外に出るにも熱中症が心配．夜も気温がなかなか下がらず熱帯夜の状態である．こんな日には一杯のビールがうまい!(島)

●ブラジルゴリンがぶじ始まった．BRAサッカーはやっと監督交替して臨むも消極病がまだみえる．しかし屋外競技を冷房の部屋で見ているだけだが，屋外の東京の猛暑に触れると選手達の厳しさが湧いてくる．(ち)

Interface

©CQ出版(株) 2016　振替　00100-7-10665
2016年10月号　第42巻　第10号(通巻第472号)
2016年10月1日発行(毎月1日発行)
定価は裏表紙に表示してあります

発行人／寺前 裕司　編集人／上村 剛士
編集者／村上 真紀　西野 直樹　野村 英樹
島田 義人　中澤 里美
デザイン・DTP／クニメディア株式会社
表紙デザイン／株式会社コイグラフィー
表紙・本文イラスト／神崎 真理子
本文イラスト／米田 裕　浅井 亮八
広告／千葉 昌之　菅原 利江
発行所／CQ出版株式会社
〒112-8619　東京都文京区千石4-29-14

電話／編集 03-5395-2122　FAX／03-5395-2127
広告 03-5395-2131
販売 03-5395-2141　その他 03-5395-2115
URL http://interface.cqpub.co.jp/
E-mail supportinter@cqpub.co.jp
CQ Publishing Co.,Ltd.／4-29-14 Sengoku, Bunkyo-ku, Tokyo 112-8619, Japan
印刷製本／大日本印刷株式会社


日本ABC協会加盟誌
(新聞雑誌部数公査機構)　ISSN0387-9569

編集者募集！（正社員）

仕事は基本やり放題です．
自分のアイディアを形にして後世に残る本を作りませんか．

職　種　月刊誌および関連書籍の企画・編集
資　格　電子回路設計とか開発とか経験あり．
募集人員　トランジスタ技術，インターフェース各1名
給　与　年齢と現給を考慮のうえ，当社規定により優遇
待　遇　年俸契約に基づく給与，交通費全額支給．試用期間あり
勤務地　東京都文京区千石（最寄駅：巣鴨駅）

休　日　週休2日制，年末年始，夏期
選考方法　作文，筆記，面接
提出書類　履歴書（写真添付），職歴書
書類提出先　〒112-8619　東京都文京区千石4-29-14
CQ出版㈱　三澤　宛　Tel.：(03)5395-2111（総務部）
E-mail：misawa@cqpub.co.jp

古株編集者の毎日

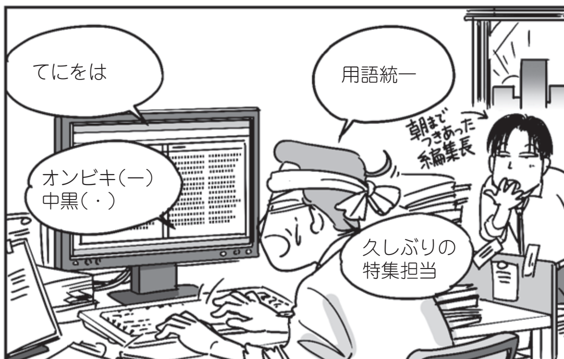




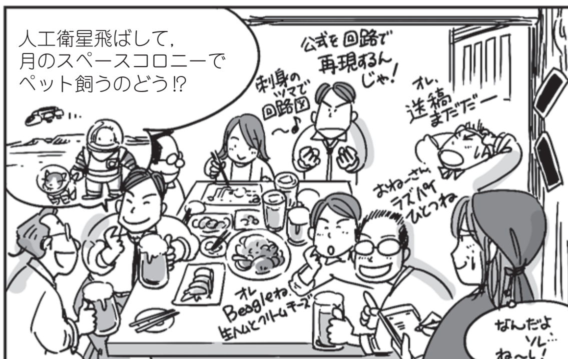
クセ1　夜な夜な六本木に繰り出し原稿依頼



クセ2　独り言でテンションを上げる



クセ3　時間を忘れて朝6時



クセ4　飲みながらワイワイ企画会議

姉妹誌

http://fpga.cqpub.co.jp/

FPGAマガジン No.14

FPGAマガジン編集部 編
B5判　144ページ
定価：2,200円＋税

特集　知らないと乗り遅れる！ XilinxもAlteraも無償時代！ 最新C開発ツール大研究

特集では，Xilinx社が無償で提供しているC開発が可能な最新開発環境Vivado HLSを使って，実際にC言語で記述した処理を高位合成してFPGAに実装するまでを解説します．処理速度を上げるためのパイプライン化までC記述で行ってみます．また，Altera社が本来有償で提供している同様のOpenCL言語開発環境Altera SDK for OpenCLを，本特集用に特別に無償で提供してもらいましたので，実験方法を紹介します．





読者プレゼント

http://interface.cqpub.co.jp/

本誌Webページの読者アンケートにご回答いただいた方の中から抽選で，下記をプレゼントいたします．アンケート結果は，本誌の誌面作りのための貴重な資料として活用させていただきます．ご協力よろしくお願いします．なお，当選者の発表は発送をもって代えさせていただきます．

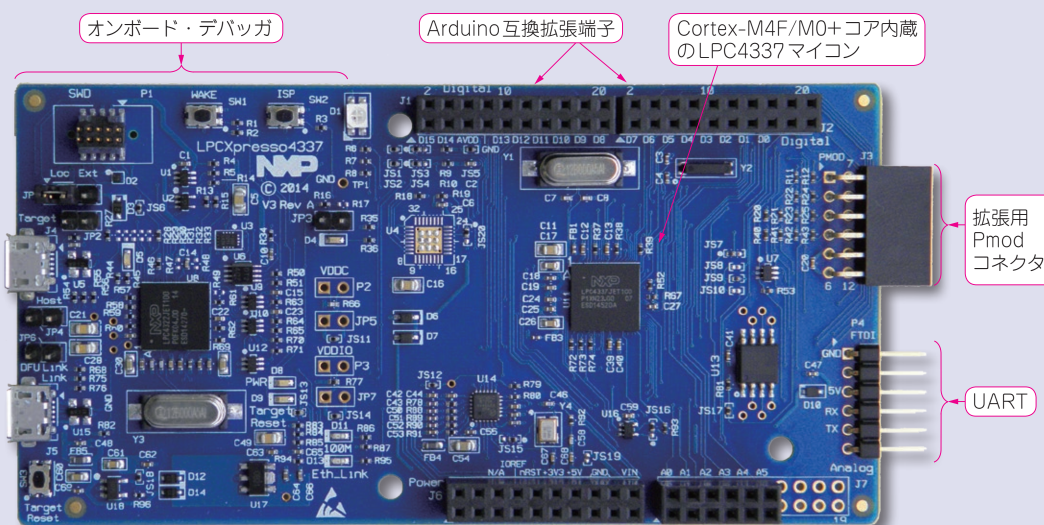
●応募締め切り　2016年9月24日

最高204MHz動作！ 高性能Cortex-M4Fマイコン LPC4337評価ボード（3名様）

LPC4337は，ARM Cortex-M4F（DSP＋FPU）とCortex-M0+を搭載するデュアルコア・マイコンです．M0+でペリフェラルのコントロールを，M4Fで演算を行えます．

M4FをスリープさせM0+のみで制御すれば，待機時消費電力を削減できます．動作周波数はM4F/ M0+ともに最高204MHzです．1Mバイトのフラッシュ・メモリを搭載します．

（提供：NXPセミコンダクターズ）



ADVERTISERS' INDEX—製品別目次

EtherCAT対応製品も取り揃え

Cortex-Mコア搭載で産業分野に最適なインフィニオンのXMC



インフィニオン テクノロジーズ ジャパン株式会社(以下 インフィニオン)は欧州を代表する半導体メーカであり，特に車載や産業など高信頼性が要求されるマイコン分野で定評がある．RISC/MCU/DSPを統合した32ビットTriCoreなど高性能のオリジナル・コアで知られているが,2012年にCortex-M4FベースのXMC4000,2013年にCortex-M0ベースのXMC1000を発表して，ARMマイコン市場に参入した．

XMCファミリは，これまで同社が培ってきた高信頼性を特徴として，産業分野からハイエンドの民生分野までカバーするユニークなARMマイコンだ．このうち次世代産業用バスの本命と言われるEtherCAT対応製品を中心に紹介しよう．お話しは，インフィニオンのパワーマネジメント&マルチマーケット事業本部 滝澤 靖明 氏と川上 彰 氏に伺った．

執筆：宮崎 仁

1．産業用途のユニークなARMマイコン

インフィニオンはマイコン・メーカとして30年以上の経験をもち，車載，産業分野向けの高信頼製品を得意とする．この分野はライフ・サイクルが長く，長期間の安定供給を求められる．これまではオリジナル・コアを中心に製品化してきたが，最近では業界標準のアーキテクチャを求めるユーザの声も多いのだろう．それに応えるのが，Cortex-M4Fコア搭載のXMC 4000とCortex-M0コア搭載のXMC1000だ．

Cortex-Mベースのマイコンは数多く存在するが，XMCファミリは最低15年間の長期供給保証，T_A = −40〜125℃の広い動作温度範囲(XMC4xxx-K)，T_J = 110℃で20年間連続稼動可能な高信頼性，VDDP = 1.8〜5.5Vの広い電源電圧範囲(XMC1xxx)など産業分野に最適化しているのが大きな特長だ．さらに，コア仕様や組み合わせるペリフェラルなどによって品種ごとの差別化をはかり，幅広い産業機器，業務用機器，家電/照明機器などの市場に対応している．

インフィニオンでは，XMCファミリの主な市場として，ホーム&プロフェッショナル(家電，照明，電動工具など)，ビルディング・オートメーション，パワー&エネルギ(UPS，ソーラ・インバータ，SMPSなど)，トランスポーテイション(電動バイク，建機，農機など)，ファクトリ・オートメーションの5本柱を想定している．また，マイコンが処理するアプリケーションとしてはモータ制御，照明，電力変換，通信などを想定しているという．

■ XMC1000はモータが使いやすい

Cortex-M0コア搭載のXMC1000は，ローエンドのXMC1100からハイエンドのXMC1400まで4品種をラインナップしている(**図1**)．8〜16ビット・マイコンの置き換えにも適した品種で，コア・クロック(MCLK)の最大動作周波数は32MHz(XMC1100/1200/1300)または48MHz(XMC1400)に抑えられているが，ペリフェラル・クロック(PCLK)はその2倍の周波数で動作，高性能のアプリケーションに対応できる．

特に，XMC1300は除算やsin/cos演算を高速化する専用のMath co-processorを備えており，Cortex-M3クラスのモータ制御演算能力をもつ．モータ1個の制御を小型かつ低コストで実現できるのは大きな特長だ．また，XMC1400はMath co-processorに加えて2チャネルのCAN通信を備えており，産業用途で幅広い応用が可能だろう．

■ XMC4000は2個のモータを同時制御

Cortex-M4Fコア搭載のXMC 4000は，現在のところローエンドのXMC4100からハイエンドのXMC4800まで7品種をライ

図1 XMC1000とXMC4000のラインナップ

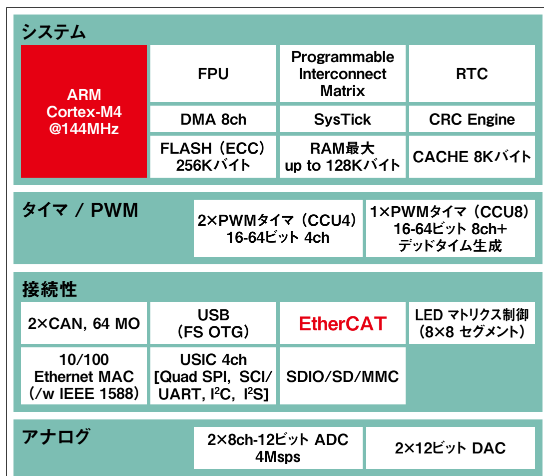
XMC4000シリーズ ARM Cortex-M4F

- XMC4100/4200：最大256KB フラッシュ，最大40KB RAM，48-64ピン
- XMC4300：256KB フラッシュ，128KB RAM，64-100ピン ＞144MHz 動作 ＞Ethernet ＞EtherCAT
- XMC4400：最大512KB フラッシュ，80KB RAM，64-100ピン ＞120MHz 動作 ＞Ethernet ＞ΔΣデモジュレータ
- XMC4500：最大1MB フラッシュ，最大160KB RAM，100-144ピン ＞外部バス ＞SDカードIF
- XMC4700：最大2MB フラッシュ，最大352KB RAM，100-196ピン ＞144MHz 動作 ＞6ch CAN
- XMC4800：最大2MB フラッシュ，最大352KB RAM，100-196ピン ＞EtherCAT

XMC1000シリーズ ARM Cortex-M0

- XMC1100：最大64KB フラッシュ，16-40ピン
- XMC1200：最大200KB フラッシュ，16-40ピン ＞9ch LED 制御(BCCU) ＞3×アナログ・コンパレータ
- XMC1300：最大200KB フラッシュ，16-40ピン ＞Math コプロセッサ ＞CCU8 PWM タイマ ＞ホール&エンコーダIF
- XMC1400：最大200KB フラッシュ，40-64ピン ＞2×CAN ＞2×CCU8 ＞最大4chシリアルIF

図2　XMC4300のブロック図



ンナップしている（**図1**）．コア・クロック（MCLK）の最大動作周波数は80MHz（XMC4100/4200），120MHz（XMC4400/4500）または144MHz（XMC4300/4700/4800）で，モータ2個の同時制御が可能だという．最大6チャネルのCANや，Ethernet，EtherCAT（XMC 4300/4800）など通信機能も充実している．

2. 次世代産業用バスの本命，EtherCATを搭載したXMC4300/4800

工場などには数多くの機器があり，それらをネットワーク化して制御/管理するための産業用バス（フィールドバス）が使われている．一言で工場といっても，化学プロセス機器，水処理機器，工作機械，搬送機器，計測機器，制御機器，検査機器などさまざまな機械があり，業種や規模によっても要求はさまざまに異なるため，フィールドバスもさまざまな規格が並立している．その中で，最近ではEthernetをベースにした高速性とリアルタイム性を両立したオープンなフィールドバス規格としてEtherCATの普及が始まっている．たとえば，今年5月にはトヨタ手自動車の工場でEtherCATを全面的に採用が決まった．

XMC4000のハイエンド品種として昨年4月に発表されたXMC4800は，フラッシュ・メモリやアナログ，PHY用クロックをすべて内蔵しているのが大きな特長であり，省スペース設計，BOMコスト（部品構成の原価）削減に大きく貢献する．さらに，今年2月にはより小型，低コストでEtherCATを実現できるXMC 4300が発表された（**図2**）．これによって，EtherCATの普及がさらに進むと考えられる．

Cortex-Mコア・マイコンでは世界初! EtherCATコントローラ内蔵



これまでEtherCATは，ドイツの制御機器メーカであるBeckhoff社が供給するASICを利用するか，FPGAと汎用マイコンを組み合わせるのが一般的だった．そのため，スレーブ側機器を十分に小型，低コスト化できないという難点があった．XMC4300，XMC

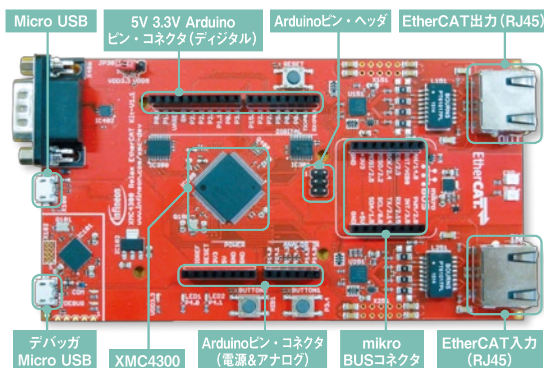


写真2　XMC4300 Relax Kit（Cortex-M4/144MHz，EtherCATノード内蔵，256Kバイト・フラッシュ，128KバイトSRAM）

4800には市場実績のあるEtherCATハードIPが搭載されたことで，開発コストも大きく軽減できる．

3. XMCファミリの開発環境

インフィニオンのマイコンは開発環境にも大きな特長があり，本格的なコード・ジェネレータであるDAVEを中核として，EclipseベースのIDE（統合開発環境）やGUIコンパイラ/デバッガ/フラッシュ・ローダを組み合わせている．DAVEは以前からインフィニオンの8/16/32ビット・マイコンで使われており，XMCファミリの登場に合わせてARMマイコンへの対応を果たした．昨年の2月にリリースされたバージョン4が最新版となるが，ローレベル・ドライバのXMCLIBやGUIベースで選択や改良ができるDAVE APPsなどのツールをもち，アプリの開発や再利用を効率よく実現できる．DAVEは，無償で提供されている．

DAVEが生成するソース・コードはGNU，ARM/Keil，IAR，Atollicなどのコンパイラ/IDEにインポートでき，ARMの豊富なエコシステムとの親和性も高く，MATLAB/Simulinkとの連携もできる．

XMC1000のブートキット・シリーズには，超小型で手軽に利用できるXMC 2Go（**写真1**）も用意されている．XMC 4000では，特長ある機能をもつRelax Kitが多数用意されている．たとえば，EtherCAT対応のXMC 4300はオンボードでEtherCATを使用できるXMC4300 Relax Kit（**写真2**）が，XMC4800はメイン・ボードとEtherCATモジュールを組み合わせて使用できるXMC4800 Relax Kitが利用できる．



写真1　XMC 2Go Kit（Cortex-M0/32MHz，64Kバイト・フラッシュ，16KバイトRAM）

読者プレゼント

・XMC4300 Relax EtherCAT Kit（**写真2**）を3名様
・XMC2GO Kit（**写真1**）を5名様
にプレゼントします．下記のサイトよりご応募ください．
https://cc.cqpub.co.jp/system/enquete_entry/523/

インフィニオン テクノロジーズ ジャパン株式会社
〒141-0032 東京都品川区大崎1-11-2 ゲートシティ大崎イーストタワー21F
TEL：03-5745-7100

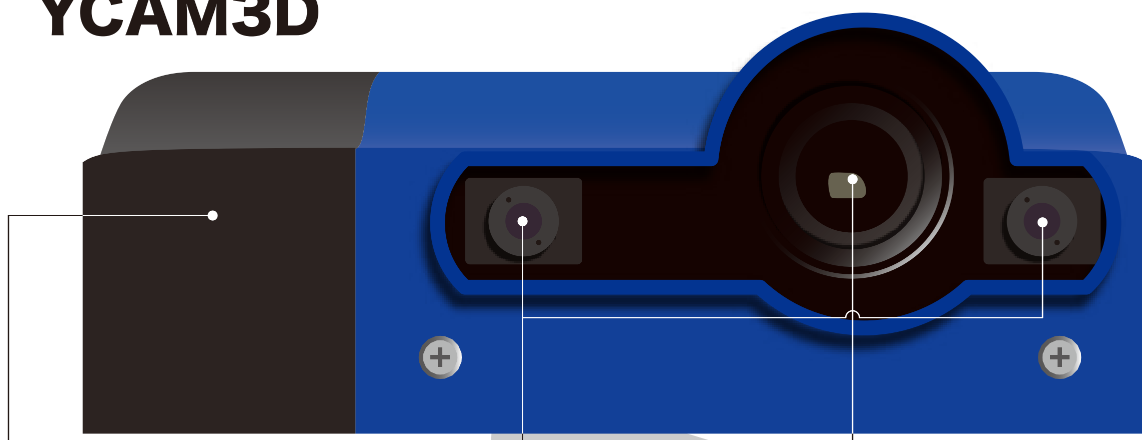
ステレオ位相シフト方式
FA用3次元画像計測カメラ

国産

【ステレオ相関方式にも対応】

ロボットピッキング、3次元計測における〈高精度〉かつ〈低価格〉なソリューションを提供します。

YCAM3D



サイズ：190mm(W)×80mm(H)×120mm(D)
重　量：900g

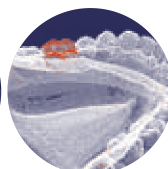
高性能小型ステレオカメラ
130万画素（1280×960pixel）
最大WD（ワーキングディスタンス）
1,000mm

FA用パターン投影プロジェクタ
青色LED（標準）
※計測対象物によりLED色を選択可能。
照射エリア 最大□800mm

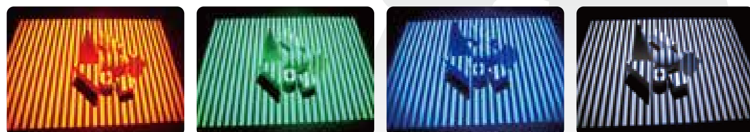
- ○ 高速撮影0.2秒
- ○ PCインターフェース USB2.0/GigE
- ○ 軽量・コンパクトだから簡単組込み
- ○ カメラレンズは撮像範囲、WDに応じて変更可能(M12マウント)
- ○ 多様な適用範囲
  寸法計測、3Dマッチングを必要とする産業、医療分野等
- ○ プロジェクタレンズは照射エリアに応じて変更可能(Cマウント)
  LED各色対応



ランダムピッキング



3次元計測
（義歯精度計測）





販売代理店
Forks
株式会社フォークス 営業部
〒110-0004 東京都台東区下谷1-11-15
TEL:03-5246-8061　FAX:03-5828-9377
E-mail:sales@yaya.forks.co.jp　http://www.forks.co.jp/

開発発売元
株式会社 YOODS
株式会社 YOODS（ユーズ）
〒754-0011 山口県山口市小郡御幸町4-9 山陽ビル小郡
TEL:083-976-0022　FAX:083-976-0023
E-mail:info_yoods@yoods.co.jp　http://www.yoods.co.jp

- ECNは，日本の電子産業を支える中小企業の製品を紹介するコーナーです．
- スタートアップ企業の事業拡大・協業も積極的に支援していきます．
- 製品の追加情報やお得情報を専用Web（http://ecn.cqpub.co.jp/）に掲載しています．



製品詳細情報の
専用Webページへ移動します。

CQ ECN 検索

ザ・低価格産業用カメラ！ DFKシリーズ　USB2/USB3/GigE



- ■ドイツThe Imaging Source社製
- ■Linux/Windows対応
- ■DirectShow対応，UVCFirmware有
- ■MATLAB, Halcon, LabVIEW, VisionPro動作
- ■SDK・簡易測定・バーコード読取ソフト有
- ■ソフト・SDKの日本語マニュアル完備
- ■39,000円～（SDK込）

(株)アルゴ　http://www.argocorp.com/
TEL 06-6339-3366　FAX 06-6339-3365　argo@argocorp.com

miniPCIe対応CameraLinkボード PIXCI EB1mini



小型PC向けのminiPCIe用フレームグラバーボード．分離式でCameraLinkコネクタを自由な場所に設置可能．CameraLink Base Configuration専用でminiPCIe フルサイズI/F採用．最大転送速度200MB/秒．キャプチャーソフトウェアも付属．別売のSDKでアプリケーション開発可能．32/64bit Windows&Linux OS対応．

(株)アルゴ　http://www.argocorp.com/
TEL 06-6339-3366　FAX 06-6339-3365　argo@argocorp.com

UVC対応 映像キャプチャユニット AV.io HD



UVC／UAC対応のVGA／DVI／HDMI信号キャプチャユニット．専用ドライバのインストールが不要でパソコンとUSBで接続すればすぐに使えます．さまざまな映像機器からの画像を取込む用途に最適です．
Skypeでの映像共有にもご利用いただけ，遠隔地との情報共有にも役立つ優れモノ．
定価￥59,800（Amazonでも発売中）

(株)アルゴ　http://www.argocorp.com/
TEL 06-6339-3366　FAX 06-6339-3365　argo@argocorp.com

DC24V電源 RS-485/RS-422 LANコンバータ LNX-003-24V



LNX-003-24Vは，LANからRS-485/RS-422を使うことができるLANコンバータです．FA環境でよく使用されるDC24V電源で動作します．LNX-001やLNX-003-24V同士と接続してトンネリングモードで接続ができます．また，TCP/UDPまたはTelnetなどにより，PCからLNX-003-24Vに直接接続して動作させることも可能です．

(有)ヒューマンデータ　http://www.hdl.co.jp/ECN1609
TEL 072-620-2002　FAX 072-620-2003

PIC16F1459で作る計測器 温度，歪，分光計測



PIC16F1459を使って　熱電対計測，歪計測，分光計測，ソーラー充電などができる計測器を製作．パソコンと組み合わせて簡易リフロー炉，ロードセルアンプ，植物の葉の状態の分光反射計測ツールとなります．

(株)ATシステム　http://www.at-system.jpn.com/
TEL 053-482-9781　FAX 053-482-9782　ito@at-system.jpn.com

業界初，あの製品を復刻！ 入手困難な半導体の復刻版開発と量産



旧いICを使っている装置の延命を目的として復刻版を開発しております．H8，SH4，M32R等のCPUコアライセンスを受けて復刻します．実績としてH8マイコン＋ゲートアレイの開発も可能です．少量～中量の長期供給を対応します．終息半導体でお困りのお客様は，お気軽にお問合せ下さい．

(株)ロジック・リサーチ　http://www.logic-research.co.jp/
TEL 092-834-8441　FAX 092-834-8442　sales@logic-research.co.jp

リモコンならDAISEN
1台からリモコン製作OK！



●キー数，外形寸法が異なる9種類を用意．
すべてプラスチックケース製．
●受信ボードも各種取り揃えています．
●１台からご注文承ります．
製作台数が少ないとあきらめていた方に
小ロット対応が可能です．
ぜひ，お問い合わせを！

(株)ダイセン電子工業　http://www.daisendenshi.com/
TEL 06-6631-5553　FAX 06-6631-6886　ddk@daisendenshi.com

世界品質をアジアから
リチウム・イオン電池　設計・製造



〈Joules Miles社製電池〉
●国産セル（主にPanasonic）を採用
●台湾にて，セルメーカーの品質管理に合格した基準で設計・製造
ご検討中のお客様向けに，電池技術に関する支援全般を行うコンサルティングサービスをご用意しております．リチウムイオン電池を初めて自社製品に採用されるご相談も承ります．

(有)オーディーエス　http://www.ods-web.co.jp/
TEL 048-593-6278　FAX 048-593-0596　info-cq@ods-web.co.jp

CompactPCI PlusIO CPUボード
PC1-GROOVE



PC1-GROOVEは1.06GHzから2.66GHzのクロック周波数に対応しているIntel Core i7プロセッサを搭載したパワフルな4HP/3UのCompactPCI CPUボードです．フロントパネルには，2つのGbジャック，2つのUSBがあり，高解像度ディスプレイ用としてVesa DPコネクタまたは従来型VGAコネクタを備えています．

(株)シーピーアイテクノロジーズ　http://www.cpi-tec.jp/ekf/pc1-groove/
TEL 045-331-9201　FAX 045-331-9203　sales@cpi-tec.com

ロジアナ機能付き小型通信アナライザ
LINEEYE　ＬＥ-１５００



電池駆動可能なエントリーモデルです
●期間限定特価 9/30まで99,000円（税抜）
●調歩同期通信や非同期PPPを測定可
●RS-232CとRS-422/485に標準対応
●有効数字4桁で任意の通信速度設定
●160種類のテストデータを送信可能
●CFカードに通信ログを連続記録
●低電圧TTL通信用拡張セットを用意

(株)ラインアイ　http://www.lineeye.co.jp/
TEL 075-693-0161　FAX 075-693-0163　info@lineeye.co.jp

外付UPS不要！ATX電源
HNSP9-520P　シリーズ



《《停電瞬停からシステムを護る》》
◆無瞬断での停電バックアップバッテリはPC内部に内蔵可能
◆ATX出力に+24Vや+48V（ATXとは絶縁）などの出力が追加可能
◆高効率（80PLUS BRONZE取得），低待機電力で消費電力を削減
◆IEC/UL/CSA60950-1を取得

(株)ニプロン　http://www.nipron.co.jp/
TEL 06-6487-0611　FAX 06-6487-2212　support@nipron.co.jp

"筑波サーキット"を疾走しよう！
CQ EVミニカート筑波レース 2016/10/9



2014年から毎年秋に千葉県・袖ヶ浦のサーキット場で開催されていた「CQ EVミニカート・レース」が，今年から憧れの”筑波サーキット”（しかも“コース2000”メイン・コース！）に場所を移して開催します．
レース時間は「30分」，あなたは何周走りますか？　自作の手巻きモータをあなたはどう巻きますか？

主催:JEVRA/CQ EVミニカート・レース研究会　http://www.cqpub.co.jp/tse/
TEL 03-5395-2160　FAX 03-5395-1255　seminar@cqpub.co.jp

急募！開発設計者の募集
①電子回路②組込み③ソフトウェア



国内トップクラスのシェアを誇る溶接機メーカーが事業拡大の為，設計者を募集．

「あなたの力を形にしてみませんか」
「私達と真のモノづくりしてみませんか」

詳しくはアマダミヤチHPへ．

(株)アマダミヤチ　http://www.amy.amada.co.jp/
TEL 04-7180-9923　FAX 04-7180-9924　amy_jinji@amada.co.jp

高速デジタイザの試作・研究開発
最先端技術5GspsADC新製品APV85G4



・4CH，5Gsps，10bit ADCボード
・ALTERA社FPGA ArriaV搭載
・FPGAでデジタル信号処理を実現
・ギガビットEthernet搭載
その他8ch，1Gsps，12bitや16ch，100Msps，16bitのADCボードなど
高速デジタイザの設計，開発なども承ります．

(株)テクノエーピー　http://www.techno-ap.com/
TEL 029-350-8011　FAX 029-352-9013　order@techno-ap.com